

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。

2009 年 5 月

AN568-1.0

はじめに

アルテラの RapidIO 相互運用性リファレンス・デザインは、アルテラの RapidIO MegaCore® ファンクションおよび Texas Instruments 社の TMS320TCI6488 Communications Infrastructure Digital Signal Processor (TI 6488 DSP または TI 6488 と表記される) 間のインタフェースのサンプルを提供します。アルテラはこのリファレンス・デザインを提供して、TI 6488 によってアルテラの RapidIO MegaCore ファンクションのインストールおよび動作について示します。このリファレンス・デザインにより、アルテラ FPGA デザインへの 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示す以外に、このデザインでは、すべてのデータ・レートとサポートされるパケット・サイズに対するリンク使用状況の測定もサポートしています。この統計情報のサポートは、3つのボーレートによる Serial RapidIO リンクを介す Stratix® IV GX デバイスから TI 6488 DSP への転送用の最適なペイロード・サイズを決定するのに役立ちます。TI6488 が 1x リンク・モードのみをサポートするため、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 をターゲットにして次の RapidIO トランザクション・タイプを開始するように、RapidIO MegaCore ファンクションに指示する
 - NWRITE
 - NWRITE_R
 - SWRITE
 - NREAD
 - Maintenance Write
 - Maintenance Read
 - Doorbell メッセージ
- 任意の 1xRapidIO MegaCore バリエーションによって FPGA をコンフィギュレーションする。包括的なデザイン・バリエーションのセットには、あらゆるデータ・レート (1.250、2.500、および 3.125 GBaud) およびあらゆるサポートされるパケット・サイズにおける 1x モードが含まれている。
- 両方向の同時トラフィックのテストをサポートする。RapidIO MegaCore ファンクションおよび TI 6488 DSP の両方によって開始されるトランザクションを含む。
- スループットの統計情報の収集に対するサポート
- 自動データ整合性チェックに対するサポート

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・デザインをインストールと実行する方法を示します。本資料では、このデザインに使用されるシステム、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、また、このデザインを用いて行ったテストの性能をレポートします。このアプリケーション・ノートには、次の項があります。

- 「概要」
- 「リファレンス・デザインの概要」
- 4 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」
- 39 ページの「性能についてまとめ」
- 41 ページの「デザインにおける制限」
- 41 ページの「参考資料」
- 42 ページの「改訂履歴」

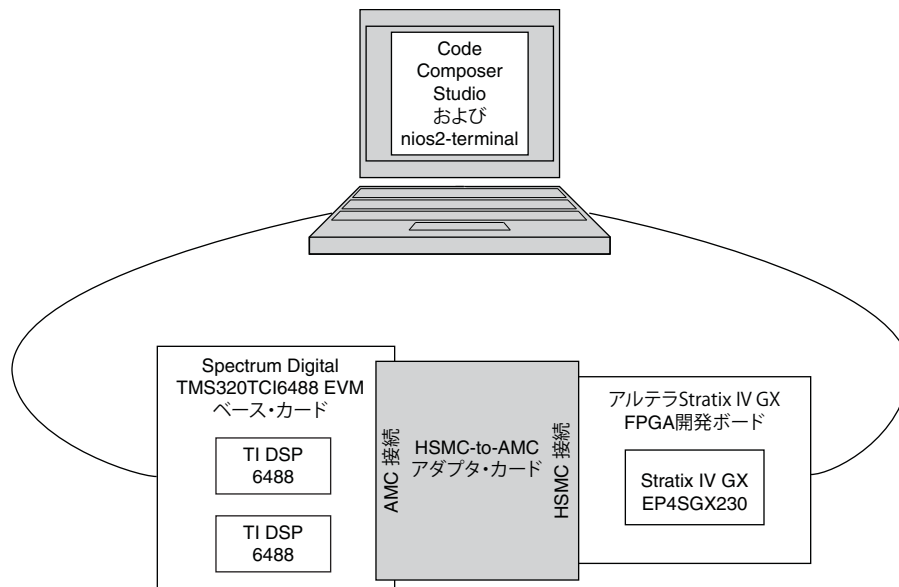
概要

このリファレンス・デザインは、アルテラの RapidIO MegaCore ファンクションをリファレンス・デザイン・インタフェース回路を介して TI 6488 DSP に接続させるサンプル・アプリケーションです。このデザインでは、アルテラ Stratix IV GX FPGA 開発ボードおよび TMS320TCI6488 評価モジュール (EVM) が使用されます。Stratix IV GX FPGA 開発ボードは Stratix IV GX EP4SGX230 デバイスを含みます。Spectrum Digital 社 (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 Code Composer Studio (CCS) 統合開発環境 (IDE) を含む TMS320TCI6488 EVM を提供しています。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したり、Stratix IV GX FPGA 開発ボード上の EP4SGX230 デバイスをプログラムするための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、完全なシステムを示します。

図 1. RapidIO 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 で動作する C コードは、以下のタスクを実行します。

- TI 6488 の内部 PLL をコンフィギュレーションする
- TI 6488 の SERDES ブロックをコンフィギュレーションする
- TI 6488 の RapidIO コマンドとステータス・レジスタ (CSR) をプログラムする
- リンク・パートナ (アルテラ FPGA) によって開始されるトランザクションに
応答する。
- リンク・パートナへのライト・トランザクションを開始し、性能を重視する。
- アルテラ FPGA への「ライトーリード」動作を開始し、データ整合性をチェックする
- アルテラ FPGA への DoorBell メッセージを開始する
- DDR2 メモリ・スペースを表示する
- RapidIO リンクのステータスをチェックする

nios2-terminal セッションでは、アルテラ FPGA において以下のタスクを実行することができます。

- RapidIO CSR をプログラムする
- 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する
- リンク・パートナによって開始されるトランザクションに
応答する。
- リンク上のデータ・スループットをモニタする
- リンクの状態をモニタする

機能の説明

RapidIO MegaCore ファンクションの各データ・レートごとに、個別のデザインが用意されています。1x 1.250 Gbaud バリエーション用のデザインは、DMA (ダイレクト・メモリ・アクセス) を用いてローカル・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。ほかの 2 つの RapidIO MegaCore ファンクション・バリエーション (2.500 および 3.125 Gbaud における 1x バリエーション) は、Avalon® Memory-Mapped (Avalon-MM) マスタ・ポートを有するパケット・ジェネレータを用いて I/O バーストを生成して、それらの I/O バーストを RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに送信します。

デザインが DMA 手法を使用してライン・レート・トラフィックによってリンクを飽和状態にするのは、1x 1.250 Gbaud のデザイン・バリエーションのみにできます。パケット・ジェネレータ手法により、デザインは RapidIO プロトコルのほかのバリエーションに対してリンクを飽和状態にすることができます。

次の項では、DMA ベースおよびパケット・ジェネレータ・ベースのバージョンのデザインについて説明し、すべてのデザイン・バリエーション内の TI 6488 DSP ファンクションおよびクロックについても記載されています。

FPGA デバイスに実装される DMA ベースの RapidIO デザイン

図 2 は、アルテラ FPGA に実装される DMA ベースの RapidIO 相互運用性リファレンス・デザインのトップレベル・ブロック図です。

図 2. DMA ベースのリファレンス・デザインのブロック図

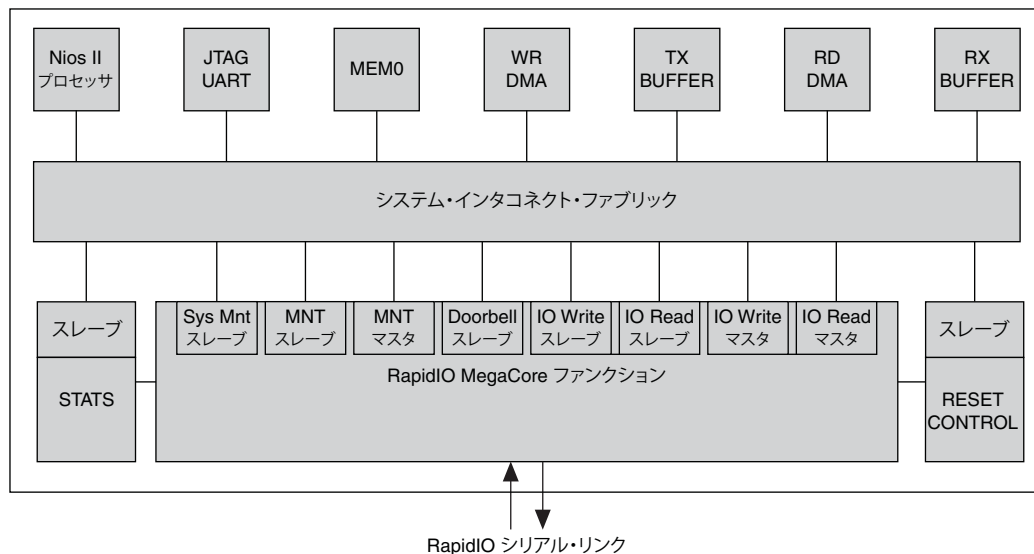


図 2 でのシステムは SOPC Builder を使用してデザインされたものです。次の項では、このリファレンス・デザイン内の主要なシステム・コンポーネントの役割について説明します。

Nios II エンベデッド・プロセッサ

FPGA デバイスに Nios® II エンベデッド・プロセッサを実装することは可能です。プロセッサは、このリファレンス・デザインに含まれる RapidIO ドライバ・ソフトウェアを実行します。このドライバにより、RapidIO MegaCore ファンクションをプログラムしたり、RapidIO MegaCore ファンクション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・リンク・パートナーにトランザクションを送受信するように 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で、FPGA に nios2-terminal と通信するメカニズムを提供します。nios2-terminal は RapidIO ドライバ用のユーザー・インタフェースです。

MEM0 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・デザインの場合、そのプログラム・コードは RapidIO のドライバです。プログラムがコンパイルされた後、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・ポートに転送しようとするデータを格納します。WR DMA コンポーネントは、このデータを異なる RapidIO ライト・トランザクションのペイロードに使用します。

RD DMA

この DMA コンポーネントは、RapidIO MegaCore ファンクション I/O ライト Avalon-MM スレーブ・ポートから RX BUFFER メモリに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って開始される 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・プロセッシング・エンドポイントまたはリンク・パートナーからのデータ・リードを受信して、そのデータを 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・プロセッシング・エンドポイントまたはリンク・パートナーからのデータ・リードを格納します。RD DMA コンポーネントはデータをこのメモリに格納します。

また、このメモリはリモート・プロセッシング・エンドポイントによってアドレスリモート・プロセッシング・エンドポイントはこのメモリ・セグメントにライト/リードすることができます。RX BUFFER メモリ・セグメントは、リモート・プロセッシング・エンドポイントによって開始され、FPGA をターゲットにする下記のトランザクションを処理します。

- NWRITE
- SWRITE

- NWRITE_R
- NREAD

このバッファには 64K バイトのアドレス指定可能なメモリがあります。

STATS

このカスタム SOPC Builder コンポーネントは次の統計情報を維持します。

- TX スループット
- RX スループット
- 送信したパケット
- 送信したバイト
- 受信したパケット
- 受信したバイト
- 拒否されたパケット
- キャンセルされたパケット
- リトライされたパケット
- CRC エラーのあるパケット
- トランスポート層によって破棄されたパケット
- シンボル・エラー
- キャラクタ・エラー

このコンポーネントを、サンプル・ウィンドウの期限が切れるたびに割り込みをアサートするようにプログラムすることができます。この割り込みに応じて、ドライバは統計情報カウンタ読み出して `nios2-terminal` セッションで表示することができます。



1x 1.250 Gbaud の RapidIO MegaCore バリエーションでは、双方向データ整合性テストを実行しながら統計情報を動的に表示することができません。この RapidIO バリエーションで実行する双方向データ整合性テストについて統計情報を表示するには、テスト終了後に `dit_stats` を実行する必要があります。

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 ファンクションに対してソフト・リセットを実行することができます。

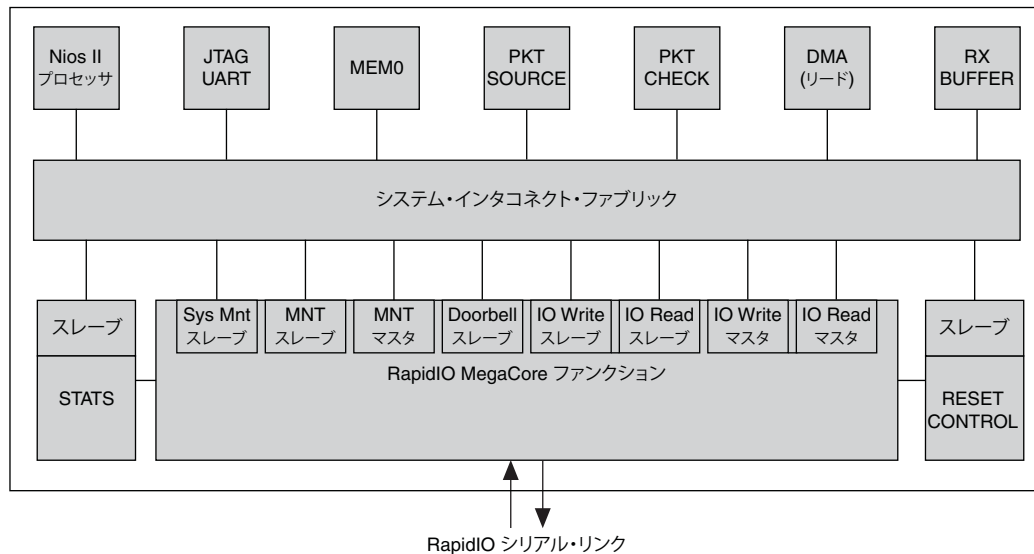
RapidIO MegaCore ファンクション

RapidIO MegaCore ファンクションはこのリファレンス・デザインの主要なコンポーネントです。このコンポーネントはリンク・パートナーに RapidIO リンクを確立します。また、RapidIO MegaCore ファンクションは、Avalon-MM インタフェース上に渡したトランザクションを対応する RapidIO トランザクションに変換して、RapidIO シリアル・リンクを介して転送します。RapidIO MegaCore ファンクションは、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、これらのバースト転送を対応する Avalon-MM スレーブまたはマスタ・ポートに渡します。

FPGA デバイスに実装されるパケット・ベースの RapidIO デザイン

図 3 は、FPGA デザインに実装されるパケット・ジェネレータ・ベースの RapidIO 相互運用性リファレンス・デザインの高レベル・ブロック図です。このバージョンのデザインはカスタムな Avalon-MM トラフィック・ジェネレータをベースとします。

図 3. パケット・ジェネレータ・ベースのリファレンス・デザインの高レベル・ブロック図



pkt_source および pkt_check コンポーネントは、パケット・ジェネレータ・ベースのデザインにありますが、DMA ベースのデザインにはありません。その他のシステム・コンポーネントは、DMA ベースでもパケット・ジェネレータ・ベースのバージョンのデザインでも同様に動作します。

pkt_source コンポーネントは、I/O バースト転送を生成可能な Avalon-MM マスタ・ポートを有するカスタム SOPC Builder コンポーネントです。このデザインでは、これらの転送は RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに送られます。

データ整合性テスト中に、pkt_source コンポーネントは、送信したライト・パケットごとに、pkt_check コンポーネント内のパケット・ディスクリプタ FIFO にパケット・ディスクリプタを書き込みます。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、取得されるリード・データを予期されるデータと比較をします。

pkt_check コンポーネントは、RapidIO MegaCore I/O リード Avalon-MM スレーブ・ポートにバースト・リードを生成可能な Avalon-MM マスタ・ポートを有するカスタム SOPC Builder 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 があります。pkt_check コンポーネントは、この FIFO が空でないことを検出すると、FIFO からパケット・ディスクリプタを読み出して、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。このリード・バーストは、パケット・ディスクリプタが指定するアドレスとデータ長をターゲットにします。リード・データを受信するとき、pkt_check コンポーネントはデータ整合性チェックを実行します。このチェックでは、取得したリード・データをパケット・ディスクリプタ・データと比較して、そして FIFO から次のパケット・ディスクリプタを読み出します。

TI 6488 DSP ファンクション

アルテラ RapidIO Megacore によって開始される NWRITE、NWRITE_R、SWRITE、および NREAD トランザクションは、TI DSP カード上の TI 6488 DSP に接続される DDR2 SDRAM をターゲットにします。DDR2 SDRAM のアドレス・スペースはアドレス 0x80000000 からスタートします。データ整合性テストはこのアドレス・スペースに書き込み、そして 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、システムにまたがってデータ整合性をチェックします。

データが DDR2 SDRAM に書き込まれた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 以降の内容を読み出します。

クロック

表 1 に、Avalon システム・クロック・レートのデータ・レート、モード、およびトランシーバ・基準クロックに対する関係を示します。

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

データ・レート (Gbaud)	モード	ALT4GXB 基準クロック・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、Stratix IV GX FPGA 開発ボード上に使用可能なクロックによって決定されます。125 MHz のクロックは、あらゆるデザイン・バリエーションの基準クロックです。

リファレンス・デザインの使用

次の項では、リファレンス・デザインのセットアップおよび使用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」
- 14 ページの「アプリケーションを実行する準備」

ハードウェア要件

リファレンス・デザインのアプリケーショ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・システムを搭載したコンピュータ。このコンピュータは Code Composer Studio IDE v3.3 および nios2-terminal セッションを同時に実行できる必要があります。
- アルテラ Stratix IV GX FPGA 開発ボード (アルテラ EP4SGX230 デバイスを含む)
- Spectrum Digital Dual 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

39 ページの「性能についてまとめ」でレポートされている性能結果は、アセンブリ版の 509310 Rev D Dual TMS320TCI6488 メザニン・ボードおよび Rev A Stratix IV 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 カードに接続される Stratix IV GX FPGA 開発ボードを示します。

図 4. アルテラ Stratix IV GX FPGA 開発ボードおよび Spectrum Digital TMS320TCI6488 AMC メザニン・ボード

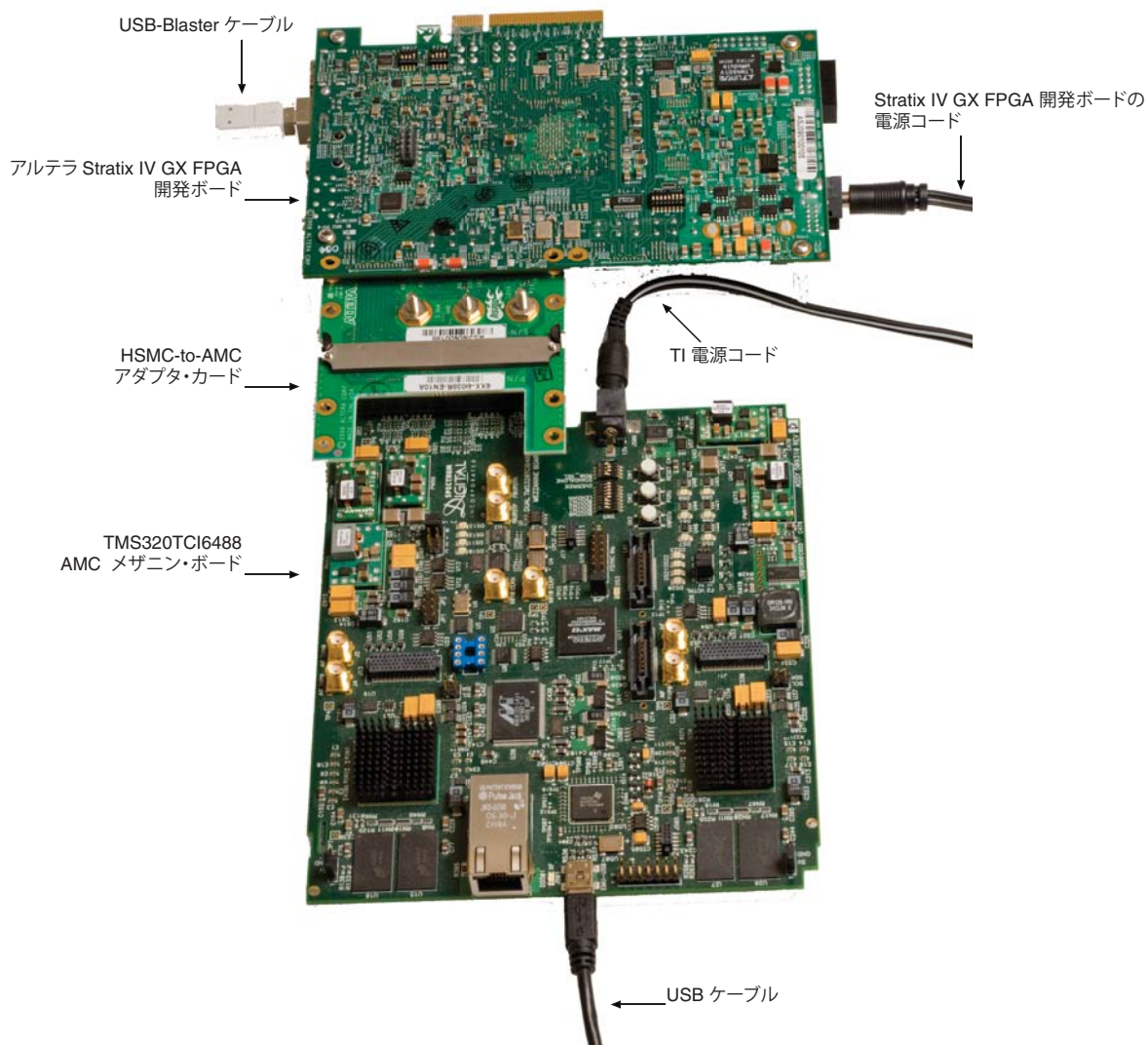


図 4 には、次のコネクタも示されています。

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

アルテラ USB-Blaster ダウンロード・ケーブルは、Stratix IV GX FPGA 開発ボード上の FPGA のコンフィギュレーション、および nios2-terminal セッションと FPGA 間の通信に使用されます。nios2-terminal 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標準の入力、出力、およびエラー位置 (stdio) です。nios2-terminal セッションは RapidIO ドライバ用のユーザー・インタフェースです。

Code Composer Studio IDE は USB を介して TI DSP カードと通信します。TI 12 V 電源コードは TI DSP カードに電源を提供し、16 V 電源コードは Stratix IV GX FPGA 開発ボードに電源を提供します。

ソフトウェア要件

このリファレンス・デザインのアプリケーションには次のソフトウェアが必要です。

- アルテラ Quartus® II v9.0 ソフトウェア (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 Nios II Embedded Design Suite (EDS) v9.0
- 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について説明します。

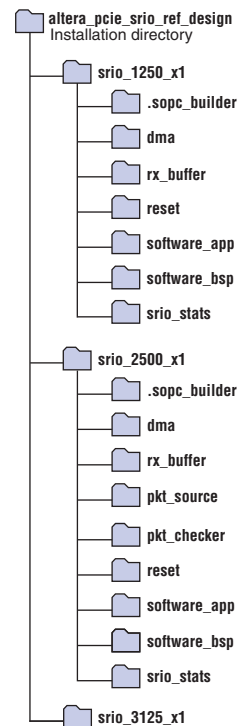
パッケージのダウンロード

このリファレンス・デザインに必要なすべてのファイルは **altera_6488_srio_ref_design.zip** ファイルに含まれています。このファイルは [Serial RapidIO To TI 6488 DSP Reference Design](#) ウェブページ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

altera_6488_srio_ref_design.zip ファイルをこのプロジェクト用の作業ディレクトリに解凍します。ファイル解凍後、作業ディレクトリには **ltera_pcie_srio_ref_design** および **altera_ti_srio_ref_design** という 2 つのサブディレクトリがあります。**ltera_pcie_srio_ref_design** ディレクトリにはアルテラ FPGA をプログラムするためのファイルがあります。図 5 に、**altera_pcie_srio_ref_design** ディレクトリの構造を示します。

図 5. altera_pcie_srio_ref_design ディレクトリの構造



パッケージ内容の説明

altera_pcie_srio_ref_design ディレクトリには、**srio_<rate>_<mode>** (**srio_1250_x1**、**srio_2500_x1**、**srio_3125_x1**) の3つのサブディレクトリがあります。

srio_1250_x1 ディレクトリには、1.250 Gbaud 1x RapidIO MegaCore ファンクション用の FPGA デザインがあります。このデザインでは、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礎となります。

srio_2500_x1 および **srio_3125_x1** ディレクトリには残りの RapidIO MegaCore ファンクション・バリエーション、すなわち 2.500 および 3.125 Gbaud における 1x バリエーションが含まれています。これらのデザインでは、Avalon-MM マスタ・ポートを有するパケット・ジェネレータを用いて I/O バースト転送を生成し、それらを RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに送信します。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礎となります。このデザインでは、DMA 手法を使用してライン・レート・トラフィックによってリンクを飽和状態にすることはできません。パケット・ジェネレータ手法により、デザインはこのモードおよびこれらのレートにおける RapidIO プロトコルに対して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、該当するデザインによってアルテラ FPGA をプログラムするための SRAM オブジェクト・ファイル (**.sof**) が含まれています。また、これらのディレクトリにはそれぞれ、下記のとおり、完全なデザインを再生成するために必要なすべてのファイルが含くまれています。

- **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 デザイン用のトップレベル・ラッパーです。このファイル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、8 ページの「クロック」を参照してください。

各 **srio_<rate>_<mode>** ディレクトリには、下記の関連するサブディレクトリ (DMA ベースまたはパケット・ジェネレータ・ベースのデザイン) が含まれています。

表 2. **srio_<rate>_<mode>** ディレクトリにあるサブディレクトリ (1 / 2)

ディレクトリ名	DMA ベース / パケット・ベース	説明
.sopc_builder	両方	SOPC Builder 用の Peripheral Template ファイル (.ptf) が含まれています。これらのファイルは SOPC Builder にある使用可能なコンポーネントを記載します。
dma	両方	このリファレンス・デザインで使用される DMA コントローラ用の Verilog コードが含まれています。また、コンポーネントの機能およびパラメータを説明する SOPC Builder 用 TCL ファイルも含まれています。
rx_buffer	両方	Avalon-MM トラフィック・シンク用の Verilog HDL コードおよび SOPC Builder 用の TCL ファイルが含まれています。
pkt_source	パケット・ベース	Avalon-MM トラフィック・ジェネレータ用の Verilog HDL コードおよび SOPC Builder 用の TCL ファイルが含まれています。
pkt_checker	パケット・ベース	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などのデータ整合性チェック用の Verilog HDL コードおよび SOPC Builder 用の TCL ファイルが含まれています。
reset	両方	RESET CONTROL ブロック用の Verilog HDL コードおよび SOPC Builder 用の TCL ファイルが含まれています。
software_app	両方	次の 3 つのファイルが含まれています。 <ul style="list-style-type: none"> ■ create-this-app – ドライバのコンパイルに使用されるスクリプト ■ srio_main_full.c – RapidIO ドライバを実装する C コード ■ srio_regs.h – RapidIO MegaCore 内部レジスタ用のニーモニック

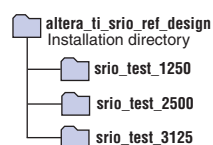
表 2. srio_<rate>_<mode> ディレクトリにあるサブディレクトリ (2 / 2)

ディレクトリ名	DMA ベース / パケット・ベース	説明
software_bsp	両方	create-this-bsp ファイルが含まれています。アプリケーションに使用される HAL ドライバ (この場合には RapidIO ドライバ) を作成するスクリプトです。
srio_stats	両方	STATS コンポーネント用の Verilog HDL コードおよび SOPC Builder 用の 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](#)」で説明したようにファレンス・デザインの zip ファイルをダウンロードして、そしてファイルを解凍した後、作業ディレクトリに **altera_ti_srio_ref_design** サブディレクトリが現れます。このディレクトリには TI 6488 DSP をプログラムするためのファイルが含まれています。図 6 に、**altera_ti_srio_ref_design** ディレクトリの構造を示します。

図 6. altera_ti_srio_ref_design ディレクトリの構造



パッケージ内容の説明

altera_ti_srio_ref_design ディレクトリには、シリアル RapidIO データ・レートごとに 1 つのディレクトリ、合計 3 つのサブディレクトリがあります。

各ディレクトリには次のファイルがあります。

- **lnk.cmd**
- **main_unidirectional.c**
- **main_bidirectional_perf.c**
- **main_bidirectional_dit.c**
- **main_tx_dbell.c**
- **srio_test_<rate>.pjt**

表 3 に、これらのファイルについて説明します。

表 3. altera_ti_srio_ref_design サブディレクトリ内のファイル

ファイル	説明
lnk.cmd	TI 6488 DSP 用のリンカ・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 をプログラムするコードです。
srio_test_<rate>.pjt	Code Composer Studio プロジェクト・ファイルです。 .c ファイルのコンパイルについての関連情報が含まれています。

srio_test_<rate>.pjt ファイルは、パスをターゲットとされるライブラリに含めます。ライブラリの位置を反映するには、このファイルを編集して、パスを例 1 に示すように変更しなければなりません。

例 1. srio_test_<rate>.pjt ファイル内のパス

```
ProjectDir="C:\srio_ref_design\PCIE_a_6488\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\PCIE_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たり、ハードウェアを接続したり、アルテラ FPGA をプログラムしたり、アプリケーションを選択したり、TI 6488 DSP をプログラムしたり、選択した性能テスト、データ整合性テスト、および Doorbell メッセージ・テストを実行したりする必要があります。次の項では、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。

ハードウェアの接続

ハードウェアを接続する前に、10 ページの「ソフトウェア要件」で述べた必要なソフトウェアをインストールしなければなりません。

ハードウェアを接続するには、次の手順を実行します。

1. CCS v3.3 をインストールします。
2. EVM と同梱されている「*TMS320TCI6488 EVM Quick Start Installation Guide*」での指示に従って、TI DSP カードに電源を投入して初期化をします。
3. TI DSP カードから電源を切断します。
4. TI DSP カードから TI USB ケーブルを取り外します。
5. HSMC-to-AMC コネクタを使用して、Stratix IV GX FPGA 開発ボードを TI DSP カードに接続します。
6. TI DSP カードに TI USB ケーブルを再接続します。
7. USB-Blaster ケーブルを Stratix IV GX FPGA 開発ボードに接続します。
8. Stratix IV GX FPGA 開発ボードに電源を投入します。
9. TI DSP カードに電源を投入します。
10. Stratix IV GX FPGA 開発ボードをオンにします。

1x 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム

RapidIO 相互運用性リファレンス・デザインの 1x 1.250 Gbaud 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、次の手順を実行します。

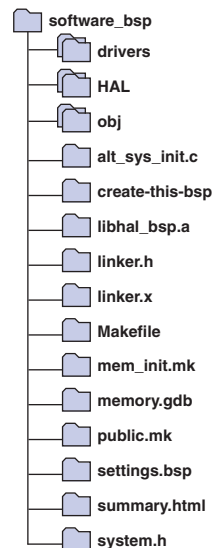
1. Windows の Start メニューから、**All Programs > Altera > Nios II EDS <version_number>** をポイントし、**Nios II <version_number> Command Shell** をクリックして Nios II コマンド・シェルを起動します。
2. **altera_pcie_srio_ref_design\srio_1250_x1** ディレクトリに移動します。

3. **software_bsp** ディレクトリに移動します。 **create-this-bsp** スクリプト・ファイルはこのディレクトリ内の唯一のファイルであるはずです。
4. このリファレンス・デザインに必要なすべての HAL ドライバをビルドするには、次コマンドを入力します。

```
./create-this-bsp ←
```

これで、**software_bsp** ディレクトリには、[図 7](#) に示した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。

図 7. create-this-bsp スクリプト実行後の software_bsp ディレクトリ構造



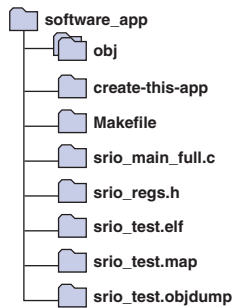
5. **..software_app** ディレクトリに移動します。
このディレクトリには下記の 3 つのファイルがあります。
 - **srio_regs.h**
 - **srio_main_full.c**
 - **create-this-app**
6. **srio_main_full.c** 内のドライバ・ソフトウェアをコンパイルするスクリプトを実行するには、次のコマンドを入力します。

```
./create-this-app ←
```

 **srio_regs.h** ファイルには、アルテラ RapidIO MegaCore ファンクション内の Serial RapidIO レジスタ用のニーモニックが含まれています。

これで、**software_app** ディレクトリには、[図 7](#) に示した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。

図 8. create-this-app スクリプト実行後の software_app ディレクトリ構造



create-this-app の実行時にコンパイル・エラーが発生しなかったら、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 のプログラム、ソフトウェア・イメージのダウンロード、および nios2-terminal セッションの起動を行うには、次の手順を実行します。

1. FPGA をプログラムするには、コマンド・シェルで次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、次のコマンドを入力します。
`nios2-download --device=1 -g srio_test.elf` ←
3. nios2-terminal セッションを起動するには、次のコマンドを入力します。
`nios2-terminal --device=1` ←

これで、アルテラ RapidIO MegaCore ファンクションは FPGA にダウンロードされました。

これで、nios2-terminal セッションが開かれており、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。

任意の RapidIO 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 を立ち上げてプログラムする必要があります。次の項ではこの手順について説明します。

残りの RapidIO デザイン・バリエーションに対してアルテラ FPGA をプログラム

RapidIO 相互運用性リファレンス・デザインのほかのバリエーションのいずれかに対してアルテラ FPGA をプログラムするには、以下の例外を除いて 14 ページの「1× 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム」での手順に従ってください。

- altera_pcie_srio_ref_design\srio_1250_x1 パスを altera_pcie_srio_ref_design\srio_<rate>_x1 パスに置き換えます。
- .sof ファイル名の srio_1250_x1.sof 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を選択

このリファレンス・デザインでは、TI 6488 DSP にロードできる 4 つのプログラムが提供されています。これらのプログラムは次の 4 つのファイルにあります。

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

main_unidirectional.c にあるプログラムは、TI 6488 DSP をスレーブのみとして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 に流れるデータの性能を測ったり、FPGA から TI 6488 DSP に渡す Doorbell メッセージをテストしたりするのに使用されます。

main_bidirectional_perf.c にあるプログラムは、TI 6488 DSP をスレーブおよびマスタの両方に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 は受信するリードとライト・トランザクションを処理し、FPGA にライト・トランザクションを開始します。

main_bidirectional_dit.c にあるプログラムは、TI 6488 DSP をスレーブおよびマスタの両方に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 は受信するリードとライト・トランザクションを処理します。また、このコンフィギュレーションは FPGA にライト・トランザクションを開始し、データ整合性チェックのためにライト位置から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状態にすることができません。したがって、このプログラムを性能測定に使用してはいけません。

main_tx_dbell.c にあるプログラムは、アルテラ FPGA 上の RapidIO コアに Doorbell メッセージを送信するように 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望ましいプログラムを指定する必要があります。リファレンス・デザインの **.zip** ファイルで提供される **srio_test_<rate>.pjt** ファイルには、4 つのプログラムをすべてソースするラインがあります。テキスト・エディタによってファイルを開いて、望ましいプログラムに関連するラインのコメントを解除し、そしてほかのラインをコメント・アウトします。

例 2 に、**main_unidirectional.c** にあるプログラムを指定するための **srio_test_<rate>.pjt** ファイル内の関連するソース・ラインを示します。

例 2. main_unidirectional.c がコンパイルされるように指定する srio_test_<rate>.pjt ファイルのセクション

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

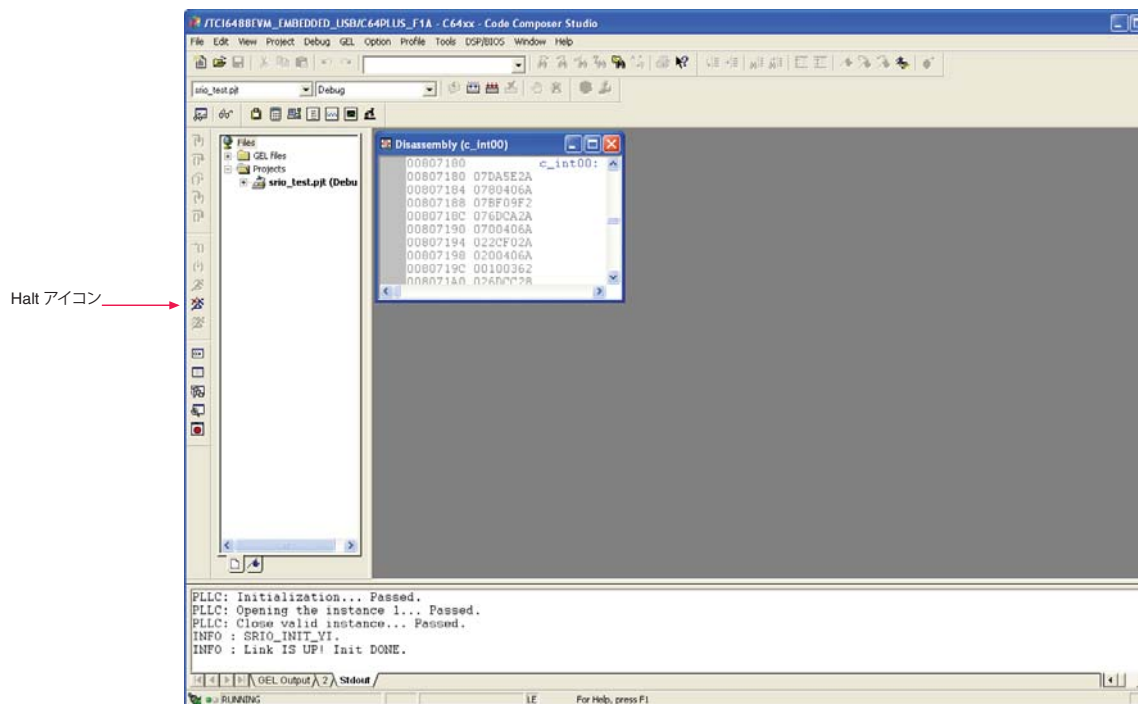
CCStudio IDE v3.3 の実行

.pjt ファイルを編集して望ましいプログラムを指定したら、TI 6488 DSP ドライバを実行することができます。この項では、任意の RapidIO バリエーションの TI 6488 DSP ドライバをロードおよび実行するために CCStudio v3.3 (CCS) セッションを開始またはリセットする方法について説明します。

新規プロジェクトを実行するように CCS IDE セッションを準備するには、次の手順を実行します。

1. TI 6488 DSP プログラムが CCS セッションで実行している場合、次の手順を実行してプログラムを中止し、前のセッションを完全に閉じます。
 - a. TMS320TCI6488 EVM の CCS ドライバのウィンドウでの左側で、**Halt** アイコンをクリックしてプログラムを中止します。図 9 に、そのアイコンを示します。
 - b. Project メニューで、**Close** をクリックします。
 - c. Debug メニューで、**Disconnect** をクリックします。
 - d. File メニューで、**Close Session** をクリックします。
 - e. Parallel Debug Manager で、File メニューから、**Exit** をクリックします。図 10 に Parallel Debug Manager を示します。

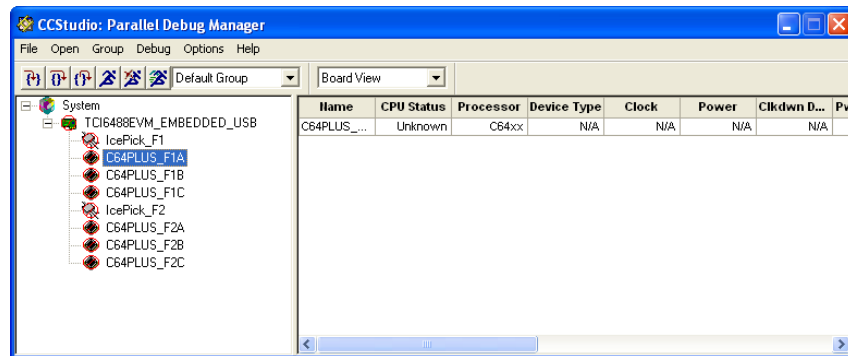
図 9. Halt アイコンがアクティブになる Code Composer Studio ウィンドウ



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** の下で、**TCI6488EVM_EMBEDDED_USB** を展開し、**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 で CCS セッションを起動します。図 10 に、この CPU を選択した後の Parallel Debug Manager を示します。
4. Debug メニューで、**Connect** をクリックします。ドライバと F1 TI 6488 DSP コア A の間に接続が確立されます。

図 10. CPU 選択後の Parallel Debug Manager



1x 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム

.pjt 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後、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、1x 1.250 Gbaud のデザイン・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について説明します。

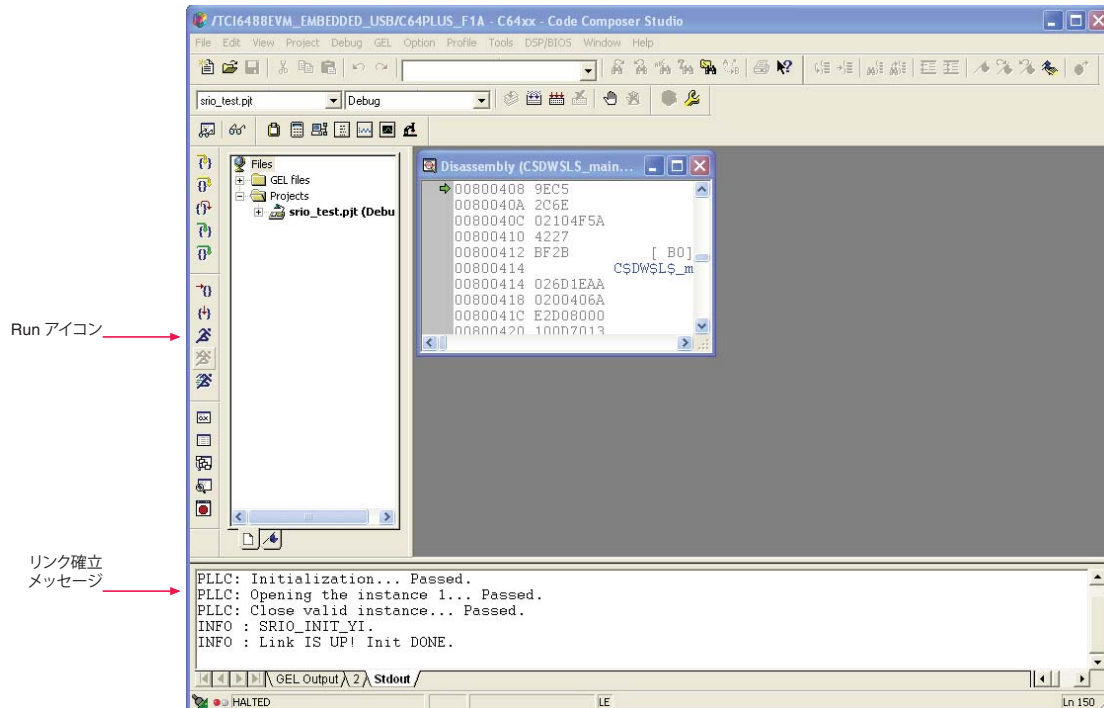
このデザイン・バリエーションをロードおよび実行するには、次の手順を実行します。

1. Project メニューで、**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. **srio_test_1250.pjt** ファイルを選択します。
4. **Open** をクリックします。
5. File View ウィンドウで、**Projects** ディレクトリから **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て、**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ためのコードがコンパイルします。
8. File メニューで、**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・ウィンドウでの左側で、**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・パネルでは、コードに埋め込まれるメッセージが表示されます。最後に表示されるメッセージは、リンクが確立したことを示します。

図 11 に、Run アイコンおよびリンクが確立したことを示す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して TI 6488 DSP をプログラム

RapidIO 相互運用性リファレンス・デザインのほかのバリエーションのいずれかに対して TI 6488 DSP をプログラムするには、以下の例外を除いて 19 ページの「1×1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」での手順に従ってください。

- **srio_test_1250** ディレクトリを **srio_test_<rate>** ディレクトリに置き換えます。
- **.pjt** ファイル名の **srio_test_1250.pjt** を **srio_test_<rate>.pjt** に置き換えます。
- **.out** ファイル名の **srio_test_1250.out** 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、選択したアプリケーションを TI 6488 DSP にロードした後、4 種類のテストの 1 つのアプリケーションを実行することができます。

DMA ベース・システム (1×1.250 Gbaud のバリエーション) とパケット・ジェネレータ・ベース・システム (残りのバリエーション) によって実装される RapidIO MegaCore バリエーションにおける性能テストおよびデータ整合性テストの実行手順はそれぞれ異なります。

次の項では、4 つのアプリケーション・プログラムを実行する方法について説明します。


単方向性能テストの実行

次の項では、単方向性能テストを実行する方法について説明します。

1× 1.250 Gbaud のデザイン・バリエーションに対して 単方向性能テストを実行

1× 1.250 Gbaud のデザイン・バリエーションに対して単方向性能テストを実行する前に、アルテラ FPGA をコンフィギュレーションして単方向性能テストをロードしていない場合は、そう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して、1× 1.250 Gbaud バリエーションに対する単方向性能テストを TI 6488 DSP にロードするには、次の手順を実行します。

1. このデザイン・バリエーションに対して FPGA をプログラムしましたが、その後に FPGA を再プログラムした場合、14 ページの「1× 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従います。

 このデザイン・バリエーションに対して FPGA をプログラムした後に FPGA を再プログラムしましたが、このバリエーションの `create-this-bsp` および `create-this-app` が実行された場合は、16 ページのステップ 1 およびステップ 2 のみを実行します。

2. `srio_test_1250.pjt` ファイルを編集して、`main_unidirectional.c` がコンパイルされるように指定します。
3. 18 ページの「CCStudio IDE v3.3 の実行」での指示に従います。
4. 19 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」での指示に従います。

単方向性能テストを実行するには、ステップ 1 で開いた `nios2-terminal` セッションに戻ります。サポートされる RapidIO ドライバ・コマンドのリストを表示するには、`h` を入力します。

次のコマンド・シーケンス（説明は表 4 を参照）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、性能をモニタします。デフォルトの `init` 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、これらのコマンドを順序立てて説明します。

表 4. 1x 1.250 Gbaud の RapidIO バリエーションにおける NWRITE 性能テスト用の nios2-terminal 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 転送を中止します。コマンド・シーケンスをこのコマンドで開始するようにして、前のテスト活動が中止されることを確保します。また、コマンド・シーケンスをこのコマンドで終了するようにして、現在のテスト活動を中止します。
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 で動作することをドライバに示します。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、および対応する Avalon-MM システム・クロック周波数の一覧については、8 ページの表 1 を参照してください。プログラムする clk 値はスループットの算出に使用されます。
pload 256	256 バイトのペイロードをバーストするように Write DMA をプログラムします。許容される値は 8 の倍数です (最小値 = 8、最大値 = 256)。
init	RapidIO MegaCore ファンクション内の CSR のサブセットを開始します。特定のプログラムされたコンフィギュレーション・レジスタについて詳しくは、srio_main_full.c ファイル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・ポートへの DMA 転送を開始します。
stats	統計情報収集の作業をイネーブルします。
z	統計情報収集の作業を中止します。

図 12 に、この RapidIO バリエーションの単方向統計情報収集のサイクルのサンプルを示します。

図 12. 1x 1.250 Gbaud における単方向統計情報収集のサイクルのサンプル

```

Nios II EDS 9.0
Baud Rate : 1250 Mbaud
TX Throughput 0.912751 Gbps TX Efficiency 91.275139%
RX Throughput 0.000000 Gbps RX Efficiency 0.000000%

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

TX Packets ..... 1495452
RX Packets ..... 0

IO S WR Bytes ..... 382835676
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・シーケンスで、start コマンドの代わりに dit コマンドを入力すると、データ整合性チェックがイネーブルされます。このテストでは、FPGA は通常の単方向性能テストと同じく、ライト・コマンドを開始します。ただし、dit テストでは、ライト動作が成功したかどうかをテストするために、FPGA はリード・コマンドも開始します。

dit テスト中に、nios2-terminal では送信したパケットおよびチェックしたパケットの数を即時に表示します。1x 1.25 Gbaud の RapidIO MegaCore バリエーションは動的な統計情報収集をサポートしません。パケットの送信およびチェックが終了したら、Serial RapidIO> プロンプトが再び表示されます。このテスト時のパケット・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チェックのコマンドに関する情報を提供します。

表 5. データ整合性テスト用の nios2-terminal コマンド

コマンド	説明
dit N	データ整合性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 進数の引数を受け取ります。データ整合性テストでは、FPGA は各パケットを書き込み、そしてそのデータを読み戻し、データが正しく書き込まれたことを確認します。
dit_stats	実行終了時に収集された統計情報を表示します。1x 1.250 Gbaud の RapidIO MegaCore バリエーションがデータ整合性テストの実行時に動的に統計情報を表示することができないため、この RapidIO バリエーションにデータ整合性テストを実行した後にこのコマンドを用いて統計情報を表示しなければなりません。

図 13 に、この RapidIO バリエーションの単方向統計情報収集のサイクルのサンプルを示します。このサンプルでは、コマンド・シーケンスでの start コマンドが dit 100 に置き換えられています。dit_stats による表示は、リード・データがデータ整合性チェックのために RapidIO MegaCore ファンクションに受信されたことを示します。

図 13. 1x 1.250 Gbaud におけるデータ整合性テストの単方向収集情報のサンプル

```

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 2.500Gbaud のデザイン・バリエーションに対して単方向性能テストを実行

1x 2.500 Gbaud のデザイン・バリエーションに対して単方向性能テストを実行する前に、アルテラ FPGA をコンフィギュレーションして単方向性能テストをロードしていない場合は、そう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して、1x 2.500 Gbaud バリエーションに対する単方向性能テストを TI 6488 DSP にロードするには、次の手順を実行します。

1. このデザイン・バリエーションに対して FPGA をプログラムしましたが、その後、FPGA を再プログラムした場合、16 ページの「残りの RapidIO デザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従います。

 このデザイン・バリエーションに対して FPGA をプログラムした後に FPGA を再プログラムしましたが、このバリエーションの create-this-bsp および create-this-app がすでに実行された場合は、16 ページの「残りの RapidIO デザイン・バリエーションに対してアルテラ FPGA をプログラム」で述べたとおり **.sof** ファイル名を置き換えて、16 ページのステップ 1 およびステップ 2 を実行します。

2. **srio_test_2500.pjt** ファイルを編集して、**main_unidirectional.c** がコンパイルされるように指定します。
3. 18 ページの「CCStudio IDE v3.3 の実行」での指示に従います。
4. 20 ページの「残りのデザイン・バリエーションに対して TI 6488 DSP をプログラム」での指示に従います。

単方向性能テストを実行するには、ステップ 1 で開いた **nios2-terminal** セッションに戻ります。サポートされる RapidIO ドライバ・コマンドのリストを表示するには、**h** を入力します。

次のコマンド・シーケンス（説明は表 6 を参照）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、性能をモニタします。デフォルトの **init** 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、これらのコマンドを順序立てて説明します。

表 6. 1× 2.500 Gbaud の RapidIO バリエーションにおける **NWRITE** 性能テスト用の **nios2-terminal** コマンド (1 / 2)

コマンド	説明
stop	パケット・ジェネレータを中止します。コマンド・シーケンスをこのコマンドで開始するようにして、前のテスト活動が中止されることを確保します。また、コマンド・シーケンスをこのコマンドで終了するようにして、現在のテスト活動を中止します。
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 が x1 RapidIO バリエーションのみサポートするため、この相互運用性デザインはモード 1 のみサポートします。
gap 4	Avalon-MM システムのクロック・サイクルの数を指定するために、Avalon-MM ライト・バーストと RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポート間のギャップを設定します。ギャップの値が小さいほど、パケット・ジェネレータはトラフィックをより短いバースト間遅延で RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに流して、スループットを増加させます。

表 6. 1x 2.500 Gbaud の RapidIO バリエーションにおける NWRITE 性能テスト用の nios2-terminal コマンド (2 / 2)

コマンド	説明
clk 75	Avalon-MM システム・クロックが 75 MHz で動作することをドライバに示します。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、および対応する Avalon-MM システム・クロック周波数の一覧については、8 ページの表 1 を参照してください。
pload 256	256 バイトのペイロードをバーストするように Write DMA をプログラムします。許容される値は 8 の倍数です (最小値 = 8、最大値 = 256)。
init	RapidIO MegaCore ファンクション内の CSR のサブセットを開始します。特定のプログラムされたコンフィギュレーション・レジスタについて詳しくは、srio_main_full.c ファイルを参照してください。
start	パケット・ジェネレータの RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートへのトラフィック転送を開始します。
stats	統計情報収集の作業をイネーブルします。
z	統計情報収集の作業を中止します。

図 14 に、この RapidIO バリエーションの単方向統計情報収集のサイクルのサンプルを示します。

図 14. 1x 2.500 Gbaud における単方向統計情報収集のサイクルのサンプル

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.825502 Gbps TX Efficiency 91.275116%
RX Throughput 0.000000 Gbps RX Efficiency 0.000000%

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

TX Packets ..... 1595149
RX Packets ..... 0

IO S WR Bytes ..... 408357988
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・シーケンスでの start コマンドを dit コマンドに置き換えると、pkt_check コンポーネントによるデータ整合性チェックがイネーブルされます。このテストでは、FPGA は通常の単方向性能テストと同じく、ライト・コマンドを開始します。ただし、dit テストでは、ライト動作が成功したかどうかをテストするために、FPGA はリード・コマンドも開始します。23 ページの表 5 に、このデータ整合性チェックに関する情報を提供します。

図 15 に、この RapidIO バリエーションの単方向統計情報収集のサイクルのサンプルを示します。このサンプルでは、dit 80000000 コマンドはコマンド・シーケンスでの stats コマンドの前に挿入されます。表示は、リード・データがデータ整合性チェックのために RapidIO MegaCore ファンクションに受信されたことを示します。

図 15. 1x 2.500 Gbaud におけるデータ整合性テストの単方向収集情報のサンプル

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.703827 Gbps TX Efficiency 85.191330%
RX Throughput 1.703827 Gbps RX Efficiency 85.191330%

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

TX Packets ..... 2977654
RX Packets ..... 1488826

IO S WR Bytes ..... 381139592
IO M RD Bytes ..... 0

IO S RD Bytes ..... 381139592
IO M WR Bytes ..... 0

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>

```

1x 3.125 Gbaud のデザイン・バリエーションに対して単方向性能テストを実行

1x 3.125 Gbaud のデザイン・バリエーションに対して単方向性能テストを実行する前に、アルテラ FPGA を正しいバリエーションにプログラムし、TI 6488 DSP を単方向性能テストおよび正しいレート設定にプログラムする必要があります。

「1x 2.500Gbaud のデザイン・バリエーションに対して単方向性能テストを実行」での指示に適切な変更を加えて、1x 3.125 Gbaud のバリエーションをプログラムおよび実行します。プログラムを実行するとき、正しい rate (3125)、mode (1)、および clk (84) の値を指定し、そして gap を指定します。このレートにおける正しい clk 設定については、8 ページの表 1 を参照してください。gap 設定はクロック・サイクルを単位として測定され、すべてのパケット・ジェネレータ・ベースのバリエーションでは同じままであることがあります。


双方向性能テストの実行

次の項では、双方向性能テストを実行する方法について説明します。

1x 1.250 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行

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。アルテラ FPGA をコンフィギュレーションして、1x 1.250 Gbaud バリエーションに対する双方向性能テストを TI 6488 DSP にロードするには、次の手順を実行します。

1. このデザイン・バリエーションに対して FPGA をプログラムしましたが、その後に FPGA を再プログラムした場合、14 ページの「1x 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従います。

 このデザイン・バリエーションに対して FPGA をプログラムした後に FPGA を再プログラムしましたが、このバリエーションの create-this-bsp および create-this-app が実行された場合は、16 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. **srio_test_1250.pjt** ファイルを編集して、**main_bidirectional_perf.c** がコンパイルされるように指定します。
3. 18 ページの「CCStudio IDE v3.3 の実行」での指示に従います。
4. 19 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」でのステップ 1～11 を実行します。

FPGA 上および TI 6488 DSP 上で双方向性能テストを実行するには、次の手順に従います。

1. ステップ 1 で開いた **nios2-terminal** 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・シーケンスを入力します。

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

3. 19 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」でのステップ 12 を実行します。(CCS ドライバのウィンドウの左側で、**Run** アイコンをクリックしてプログラムを実行します。)
4. 統計情報収集を中止するには、**nios2-terminal** で **z** を入力します。
5. 統計情報の収集および表示を再開するには、**stats** を入力します。
6. FPGA によって開始されるトラフィックを中止したい場合 (TI DSP からの要求に対する FPGA 応答は中止しない)、**stop** を入力します。

最初に、**nios2-terminal** は FPGA から TI 6488 DSP へのトラフィックのみを表示します。ステップ 3 を実行したら、トラフィックは両方向に流れ、統計情報表示では FPGA によって開始されるトランザクションおよび TI 6488 DSP によって開始されるトランザクションに対して、0 以外の数字で表示します。

図 16 に、この RapidIO バリエーションの双方向統計情報収集のサイクルのサンプルを示します。

図 16. 1× 1.250 Gbaud における双方向性能統計情報収集のサイクルのサンプル

```
Nios II EDS 9.0
Baud Rate : 1250 Mbaud
TX Throughput 0.904162 Gbps TX Efficiency 90.416153%
RX Throughput 0.911767 Gbps RX Efficiency 91.176750%
Sample Window ..... 134217227 cycles
RapidIO Payload ..... 256 bytes
TX Packets ..... 1481377
RX Packets ..... 1493839
IO S WR Bytes ..... 379232836
IO M RD Bytes ..... 0
IO S RD Bytes ..... 0
IO M WR Bytes ..... 382423012
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
```

stop を入力した後、次のコマンド・シーケンス異なるバースト・サイズ、すなわち `<payload value>` を試みることができます。

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

`<payload value>` の許容値は 22 ページの表 4 で指定されています。

1× 2.500 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行

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。アルテラ FPGA をコンフィギュレーションして、1× 2.500 Gbaud バリエーションに対する双方向性能テストを TI 6488 DSP にロードするには、次の手順を実行します。

1. このデザイン・バリエーションに対して FPGA をプログラムしましたが、その後に FPGA を再プログラムした場合、16 ページの「残りの RapidIO デザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従います。



このデザイン・バリエーションに対して FPGA をプログラムした後に FPGA を再プログラムしましたが、このバリエーションの `create-this-bsp` および `create-this-app` がすでに実行された場合は、16 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. `srio_test_2500.pjt` ファイルを編集して、`main_bidirectional_perf.c` がコンパイルされるように指定します。
3. 18 ページの「CCStudio IDE v3.3 の実行」での指示に従います。
4. 20 ページの「残りのデザイン・バリエーションに対して TI 6488 DSP をプログラム」でのステップ 1 ~ 11 を実行します。

FPGA 上および TI 6488 DSP 上で双方向性能テストを実行するには、次の手順に従います。

1. ステップ 1 で開いた `nios2-terminal` 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・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. 19 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」でのステップ 12 を実行します。(CCS ドライバのウィンドウの左側で、Run アイコンをクリックしてプログラムを実行します。)
4. 統計情報収集を中止するには、`nios2-terminal` で `z` を入力します。
5. 統計情報の収集および表示を再開するには、`stats` を入力します。
6. FPGA によって開始されるトラフィックを中止したい場合 (TI DSP からの要求に対する FPGA 応答は中止しない)、`stop` を入力します。

最初に、nios2-terminal は FPGA から TI 6488 DSP へのトラフィックのみを表示します。ステップ 3 を実行したら、トラフィックは両方向に流れ、統計情報表示では FPGA によって開始されるトランザクションおよび TI 6488 DSP によって開始されるトランザクションに対して、0 以外の数字で表示します。

図 17 に、この RapidIO バリエーションの双方向統計情報収集のサイクルのサンプルを示します。

図 17. 1x 2.500 Gbaud における双方向性能統計情報収集のサイクルのサンプル

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.808268 Gbps TX Efficiency 90.413406%
RX Throughput 1.823538 Gbps RX Efficiency 91.176895%
Sample Window ..... 134217727 cycles
RapidIO Payload ..... 256 bytes
TX Packets ..... 1580089
RX Packets ..... 1593432
IO S WR Bytes ..... 404502736
IO M RD Bytes ..... 0
IO S RD Bytes ..... 0
IO M WR Bytes ..... 407918524
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 3.125 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行

1x 3.125 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行する前に、アルテラ FPGA を正しいバリエーションにプログラムし、TI 6488 DSP を双方向性能テストおよび正しいレート設定にプログラムする必要があります。

28 ページの「1x 2.500 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行」での指示に適切な変更を加えて、1x 3.125 Gbaud のバリエーションをプログラムおよび実行します。プログラムを実行するとき、正しい rate (3125)、mode (1)、および clk (84) の値を指定し、そして gap を指定します。このレートにおける正しい clk 設定については、8 ページの表 1 を参照してください。gap 設定はクロック・サイクルを単位として測定され、すべてのパケット・ジェネレータ・ベースのバリエーションでは同じままであることがあります。


双方向データ整合性テストの実行

次の項では、双方向データ整合性テストを実行する方法について説明します。

1x 1.250 Gbaud のデザイン・バリエーションに対して双方向データ整合性テストを実行

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データ整合性テスト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して、1x 1.250 Gbaud に対する双方向データ整合性テストをロードするには、次の手順に従います。

1. このデザイン・バリエーションに対して FPGA をプログラムしましたが、その後に FPGA を再プログラムした場合、14 ページの「1x 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従います。

 このデザイン・バリエーションに対して **FPGA** をプログラムした後に **FPGA** を再プログラムしましたが、このバリエーションの `create-this-bsp` および `create-this-app` がすでに実行された場合は、16 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. **srio_test_1250.pjt** ファイルを編集して、**main_bidirectional_dit.c** がコンパイルされるように指定します。
3. 18 ページの「**CCStudio IDE v3.3 の実行**」での指示に従います。
4. 19 ページの「**1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム**」での **ステップ 1 ~ 11** を実行します。

FPGA 上および **TI 6488 DSP** 上で双方向性能テストを実行するには、次の手順に従います。

1. **ステップ 1** で開いた **nios2-terminal** 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・シーケンスを入力します。


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

3. 19 ページの「**1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム**」での **ステップ 12** を実行します。(CCS ドライバのウィンドウの左側で、**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-terminal では送信したパケットおよびチェックしたパケットの数が即時に表示されます。**1× 1.25 Gbaud** の **RapidIO MegaCore** バリエーションは動的な統計情報収集をサポートしません。**10000000** のパケットの送信およびチェックが終了したら、**Serial RapidIO>** プロンプトが再び表示されます。このテスト時のパケット・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。

10000000 以下のパケット数を `dit` コマンドの値として使用してもかまいません。**10000000** の値を使用すると、**SignalTap® II** エンベデッド・ロジック・アナライザによってパケット・アクティビティを表示することが可能です。詳細については、「**SignalTap II エンベデッド・ロジック・アナライザによって RapidIO-Avalon インタフェースを表示**」を参照してください。

 **図 18** に、次の手順を実行する後の **nios2-terminal** ウィンドウのサンプルを示します。

1. **nios2-terminal** で、次のコマンド・シーケンスを入力します。

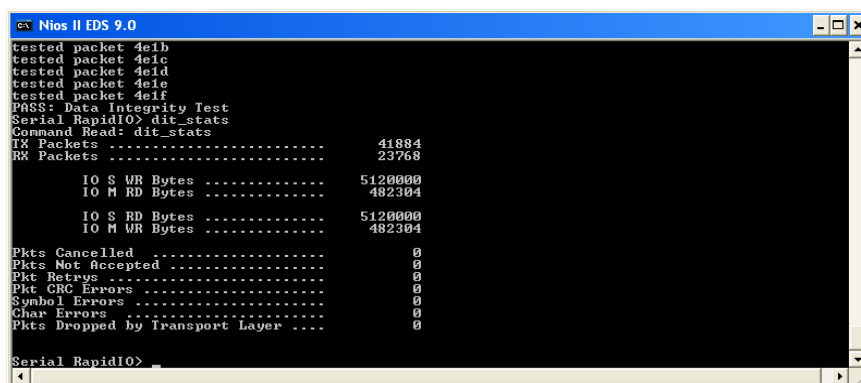
```
stop
r
rate 1250
clk 40
pload 256
init
```

2. **CCS** ドライバのウィンドウの左側で、**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-terminal で、次のコマンド・シーケンスを入力します。:

```
dit 20000
dit_stats
```

図 18. 1x 1.250 Gbaud における双方向データ整合性テストおよび dit_stats 表示のサンプル



```
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 ..... 41884
RX Packets ..... 23768
IO S WR Bytes ..... 5120000
IO M RD Bytes ..... 482304
IO S RD Bytes ..... 5120000
IO M WR Bytes ..... 482304
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>
```

図 18 では、dit コマンドを実装するために、bidirectional_dit テストはデータ整合性に対して 0x4E20 (十進数の 20000) のパケットをテストします。パケットのナンバリングは 0x0 ~ 0x4E1f です。dit_stats コマンドは、実行時に収集された統計情報を出力します。FPGA は、5120000 バイトのデータ (トランザクションごとに 256 バイトのデータを転送する 20000 件のトランザクションの総ペイロード) を書き込んで、そして読み出します。これらのデータ転送を実装するには、FPGA は 20000 個のライト・パケットおよび 20000 個のリード要求パケットを送信し、そして 20000 個のリード応答パケットを受信します。また、TI 6488 DSP はデータ整合性チェックをしながらデータの読み書きをします。統計情報のレポート時に、書き込まれたデータの整合性をチェックするために、TI DSP はすでに 1884 個のライト・パケット (合計 482304 バイトの総ペイロード) を送信し、そして 1884 個のリード要求パケットを送信しました。それに対して、FPGA は 1884 個のリード応答パケットで 482304 バイトのデータを送信しました。



ほかの 2 つの RapidIO MegaCore バリエーションとは対照的に、1x 1.250 Gbaud の RapidIO MegaCore バリエーションでは、データ整合性テストを実行しながら統計情報を動的に表示することができません。この RapidIO バリエーションで実行する双方向データ整合性テストについて統計情報を表示するには、テスト終了後に dit_stats を実行する必要があります。

1x 2.500 Gbaud のデザイン・バリエーションに対して双方向データ整合性テストを実行

1x 2.500 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行する前に、FPGA を正しいバリエーションにプログラムし、TI 6488 DSP を双方向性能テストおよび正しいレート設定にプログラムする必要があります。

Serial RapidIO> プロンプトでこのプログラムを実行する時に、次の変更を加えて、29 ページの「1x 1.250 Gbaud のデザイン・バリエーションに対して双方向データ整合性テストを実行」での指示に従って 1x 2.500 Gbaud のバリエーションをプログラムおよび実行します。

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

図 19 に、FPGA および TI 6488 DSP が 1x 2.500 Gbaud のデザイン・バリエーションにおける双方向データ整合性テストにプログラムされ、そして次の手順も実行された後の nios2-terminal ウィンドウのサンプルを示します。

1. nios2-terminal の Serial RapidIO> プロンプトで、次のコマンド・シーケンスを入力します。

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

2. CCS ドライバのウィンドウの左側で、**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. 動的な統計情報収集を表示するには、Serial RapidIO> で、次のコマンドを入力します。

```
stats ←
```

5. 動的な統計情報表示を中止するには、Serial RapidIO> で、次のコマンドを入力します。

```
z ←
```

図 19 に、stats コマンドによる出力のサンプルを示します。この図では、FPGA および TI DSP の両方によって開始されるトラフィックを示しています。

図 19. 1x 2.500 Gbaud における双方向データ整合性テストの stats 表示のサンプル

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.703830 Gbps TX Efficiency 85.191475%
RX Throughput 1.703829 Gbps RX Efficiency 85.191467%

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

TX Packets ..... 2977609
RX Packets ..... 1488878

IO S WR Bytes ..... 381127680
IO M RD Bytes ..... 12544

IO S RD Bytes ..... 381127656
IO M WR Bytes ..... 12544

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

Serial RapidIO>

```

1× 3.125 Gbaud のデザイン・バリエーションに対して双方向データ整合性テストを実行

1× 3.125 Gbaud のデザイン・バリエーションに対して双方向性能テストを実行する前に、FPGA を正しいバリエーションにプログラムし、TI 6488 DSP を双方向性能テストおよび正しいレート設定にプログラムする必要があります。

31 ページの「1× 2.500 Gbaud のデザイン・バリエーションに対して双方向データ整合性テストを実行」での指示に適切な変更を加えて、1× 3.125 Gbaud のバリエーションをプログラムおよび実行します。プログラムを実行するとき、正しい rate (3125)、mode (1)、および clk (84) の値を指定します。このレートにおける正しい clk 設定については、8 ページの表 1 を参照してください。このテストでは性能を測定していないため、gap の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、下記の 2 部分があります。

- **FPGA-to-DSP** : FPGA 上の RapidIO MegaCore ファンクションは TI DSP に Doorbell メッセージを送信し、そしてテストはそれらのメッセージの Doorbell Sent ステータスをレポートします。
- **DSP-to-FPGA** : TI DSP は、FPGA 上の RapidIO MegaCore ファンクションに Doorbell メッセージを送信し、そしてテストは RapidIO MegaCore ファンクションが受信した Doorbell Sent ステータスをレポートします。

2 部分のテストに対して、TI DSP を別々に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、アルテラ FPGA をコンフィギュレーションして、実行したい部分用の正しいプログラムを TI 6488 DSP に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 デザインのバリエーションに応じて、14 ページの「1× 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム」または 16 ページの「残りの RapidIO デザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従います。

 このデザイン・バリエーションに対して FPGA をプログラムした後に FPGA を再プログラムしましたが、このバリエーションの create-this-bsp および create-this-app が実行された場合は、16 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 の Doorbell メッセージ・テストのロードおよび実行

TI 6488 DSP に RapidIO デザイン・バリエーション用の FPGA-to-DSP Doorbell メッセージをロードするには、次の手順を実行します。

1. 適切な `srio_test_<rate>.pjt` ファイルを編集して、`main_unidirectional.c` ファイルがコンパイルされるように指定します。
2. 18 ページの「CCStudio IDE v3.3 の実行」での指示に従います。
3. RapidIO デザインのバリエーションに応じて、19 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」または 20 ページの「残りのデザイン・バリエーションに対して TI 6488 DSP をプログラム」での指示に従います。

FPGA 上および TI 6488 DSP 上で Doorbell メッセージ・テストを実行するには、次の手順に従います。

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-terminal 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・シーケンスを入力します。

```
link
init
tx_dbell_test
```

tx_dbell_test コマンドは、TI DSP に 5 グループの Doorbell メッセージ(各グループ 16 件ずつ)を送信するように FPGA 内の RapidIO MegaCore ファンクションを命令します。テストが RapidIO MegaCore ファンクションを 1 グループの Doorbell メッセージを送信するように命令した後、テストはグループ内の各 Doorbell メッセージに対してステータスおよび内容を検証します。図 20 に、FPGA 内の RapidIO MegaCore ファンクションから TI DSP に送信された第 1 グループの Doorbell メッセージの nios2-terminal へのテスト出力を示します。

図 20. FPGA から TI DSP への第 1 グループの成功した Doorbell メッセージのテスト出力

```
Nios II EDS 9.0
Serial RapidIO tx_dbell_test
Command Read: tx_dbell_test
TX Doorbells Completed 16
TX Doorbell Sent 80000
TX Doorbell Sent Status 0
TX Doorbells Completed 15
TX Doorbell Sent 80001
TX Doorbell Sent Status 0
TX Doorbells Completed 14
TX Doorbell Sent 80002
TX Doorbell Sent Status 0
TX Doorbells Completed 13
TX Doorbell Sent 80003
TX Doorbell Sent Status 0
TX Doorbells Completed 12
TX Doorbell Sent 80004
TX Doorbell Sent Status 0
TX Doorbells Completed 11
TX Doorbell Sent 80005
TX Doorbell Sent Status 0
TX Doorbells Completed 10
TX Doorbell Sent 80006
TX Doorbell Sent Status 0
TX Doorbells Completed 9
TX Doorbell Sent 80007
TX Doorbell Sent Status 0
TX Doorbells Completed 8
TX Doorbell Sent 80008
TX Doorbell Sent Status 0
TX Doorbells Completed 7
TX Doorbell Sent 80009
TX Doorbell Sent Status 0
TX Doorbells Completed 6
TX Doorbell Sent 8000a
TX Doorbell Sent Status 0
TX Doorbells Completed 5
TX Doorbell Sent 8000b
TX Doorbell Sent Status 0
TX Doorbells Completed 4
TX Doorbell Sent 8000c
TX Doorbell Sent Status 0
TX Doorbells Completed 3
TX Doorbell Sent 8000d
TX Doorbell Sent Status 0
TX Doorbells Completed 2
TX Doorbell Sent 8000e
TX Doorbell Sent Status 0
TX Doorbells Completed 1
TX Doorbell Sent 8000f
TX Doorbell Sent Status 0
```

TI DSP は 64 件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ています。したがって、第 5 グループの TI DSP の Doorbell メッセージはタイムアウトになります。図 21 に、FPGA 内の RapidIO MegaCore ファンクションによって送信した Doorbell メッセージの第 5 グループの各メッセージがタイムアウトすることを示す Doorbell Sent Status のあるテスト出力を示します。

図 21. FPGA から TI DSP への第 5 グループのタイムアウトを示す 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 の Doorbell メッセージ・テストのロードおよび実行

TI 6488 DSP に DSP-to-FPGA の Doorbell メッセージ・テストを実行して、そして FPGA 上および TI 6488 DSP 上でこのテストを実行するには、次の手順を実行します。

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-terminal セッションに戻ります。
2. Serial RapidIO> プロンプトで、次のコマンドを入力します。
init ←
3. 適切な srio_test_<rate>.pjt ファイルを編集して、main_tx_dbell.c ファイルがコンパイルされるように指定します。
4. 18 ページの「CCStudio IDE v3.3 の実行」での指示に従います。
5. RapidIO デザインのバリエーションに応じて、19 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して TI 6488 DSP をプログラム」または 20 ページの「残りのデザイン・バリエーションに対して TI 6488 DSP をプログラム」での指示に従います。

TI 6488 DSP は 16 件の Doorbell メッセージを生成し、FPGA 上の RapidIO MegaCore ファンクションに転送します。図 22 に、テスト完了後の CCS ウィンドウを示します。

6. Serial RapidIO> で、次のコマンドを入力します。
get_dbells ←

nios2-terminal では、FPGA 上の RapidIO MegaCore ファクションの受信したメッセージに関する情報を表示します (図 23 を参照)。

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

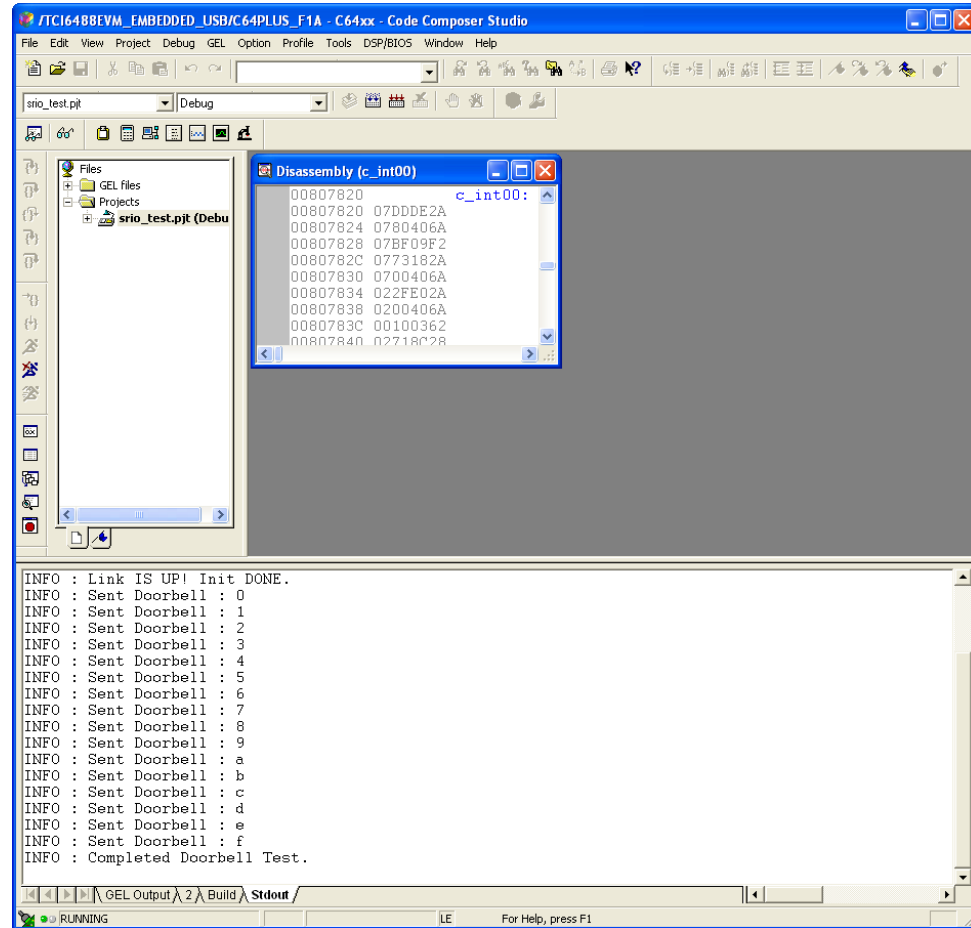
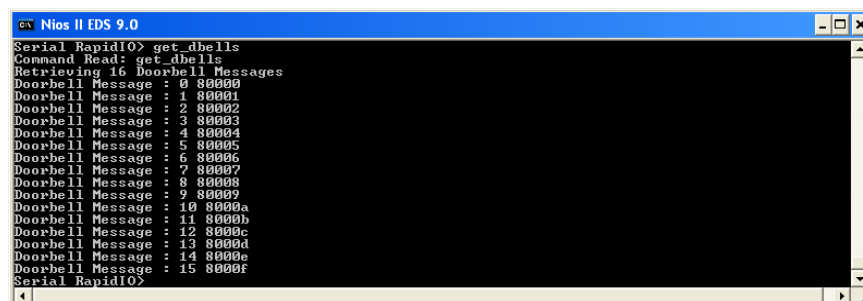



図 23. FPGA の受信した Doorbell メッセージ



SWRITE または NWRITE_R トランザクションの生成

単方向性能テスト、双方向性能テストおよび双方向データ整合性テスト用のコマンド・シーケンスは、RapidIO MegaCore ファクションに NWRITE トランザクションのみを生成させます。FPGA からの SWRITE または NWRITE_R トランザクションをテストするには、コマンド・シーケンスで、init コマンドおよび start コマンドの間に次の追加コマンドを挿入します。

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

 0x1040c レジスタおよび Input/Output Slave Mapping Window 0 Control レジスタの使用について詳しくは、[「RapidIO MegaCore Function User Guide」](#) を参照してください。

Maintenance Write トランザクションの生成

Maintenance Write トランザクションを生成するには、Serial RapidIO> で、次のコマンドを入力します。

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

ここで、<addr> は書き込み先であるリモート・プロセッシング・エンドポイント内の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、次のコマンドを入力することで、

```
rmw 0x60 0x00AAFFFF
```

TI 6488 DSP RapidIO ペリフェラルの Base Device ID が 0xAA にプログラムされます。

Maintenance Read トランザクションの生成

Maintenance Read トランザクションを生成するには、Serial RapidIO> で、次のコマンドを入力します。

```
rmr <addr>
```

ここで、<addr> は読み出されるリモート・プロセッシング・エンドポイント内の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、前のセクションでの rmw コマンドを実装した後に次のコマンドを入力することで、

```
rmr 0x60
```

0x00AAFFFF の値が返されるはずですが、

SignalTap II エンベデッド・ロジック・アナライザによって RapidIO-Avalon インタフェースを表示

このリファレンス・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** が含まれています。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 ファクション間で通信する信号のサブセットを定義し、SignalTap II エンベデッド・ロジック・アナライザ・ファイルがこれらの信号をキャプチャするように指定します。この項では、SignalTap II エンベデッド・ロジック・アナライザ・ファイルを起動して信号を表示する方法について説明します。

SignalTap II エンベデッド・ロジック・アナライザ・ファイルを使用してこれらの信号を表示するには、次の手順に従います。

1. nios2-terminal セッションを終了します。
2. 現在のデザイン・バリエーションのメイン・ディレクトリ、**altera_pcie_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 メニューで、**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** に対して **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・シェルに戻ります。
9. デザインのバリエーションに応じて、14 ページの「1× 1.250Gbaud のデザイン・バリエーションに対してアルテラ FPGA をプログラム」または 16 ページの「残りの RapidIO デザイン・バリエーションに対してアルテラ FPGA をプログラム」での指示に従って、プログラムをリロードします。
10. 次のコマンドを入力して、nios2-terminal セッションを起動します。

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

11. nios2-terminal で、21 ページの「1× 1.250 Gbaud のデザイン・バリエーションに対して 単方向性能テストを実行」での指示に従って、start コマンドを含むテスト・コマンド・シーケンス内の初期コマンドを入力します。
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。図 24 に、そのアイコンを示します。

図 24. SignalTap II エンベデッド・ロジック・アナライザの Autorun Analysis アイコン

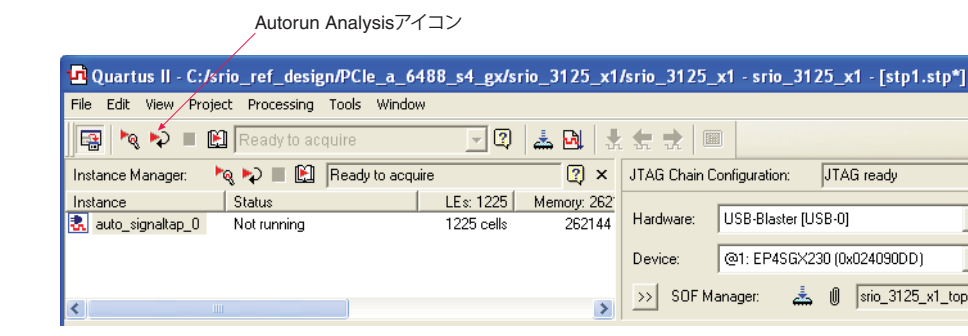
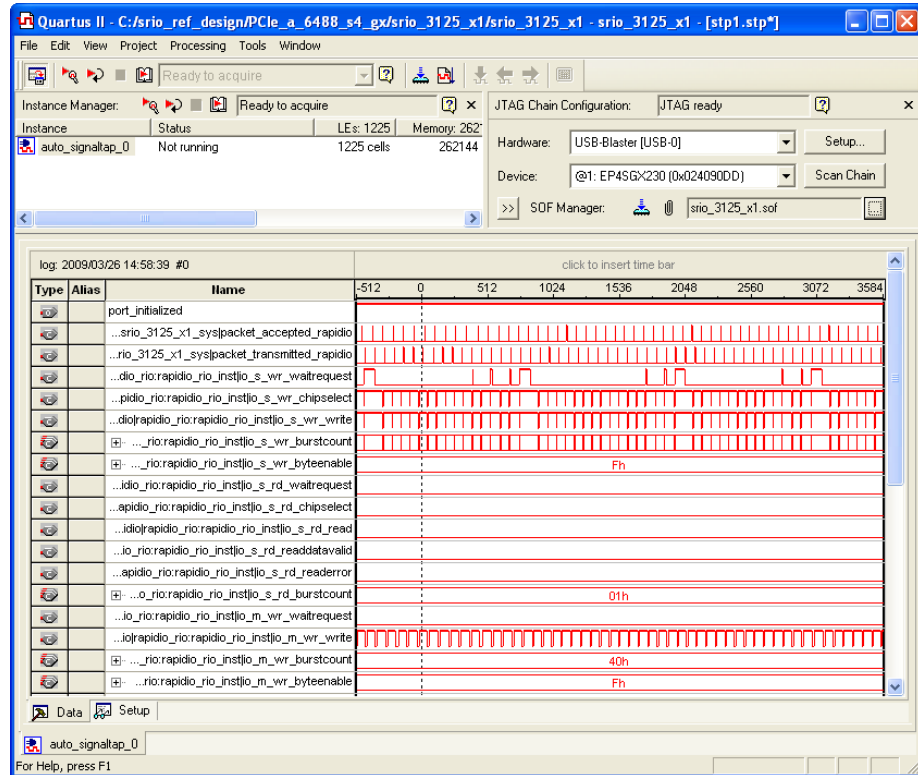


図 25 に、SignalTap II エンベデッド・ロジック・アナライザを示します。このウィンドウでは、RapidIO MegaCore ファンクションの I/O スレーブ・ライト・インタフェースおよび I/O マスタ・ライト・インタフェース上のアクティビティを表示しています。

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



性能についてまとめ

この項では、3つのデザイン・バリエーションに対する性能観測をそれぞれ表示する3つのグラフが含まれています。

この項にレポートされた性能結果は、Quartus II ソフトウェア v9.0、CCS IDE v3.3、アセンブリ版の 509310 Rev D TI DSP カード、および Stratix IV GX FPGA 開発ボード Rev A を使用して得られたものです。

図 26 ~ 図 28 のグラフでは、このデザインによって測定された性能値を示します。各グラフには、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 における x1 RapidIO MegaCore ファンクション (スループット vs. ペイロード)

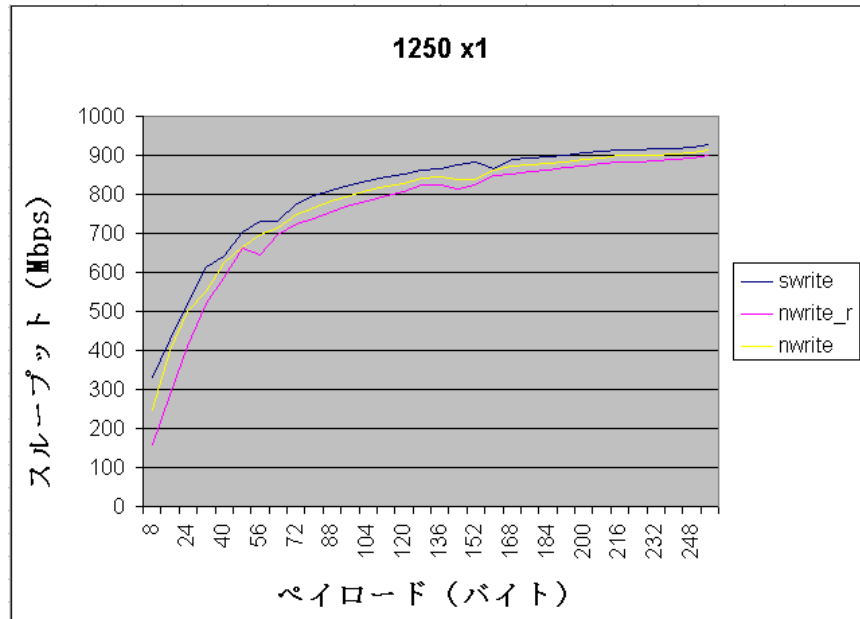


図 27. 2.500 Gbaud における x1 RapidIO MegaCore ファンクション (スループット vs. ペイロード)

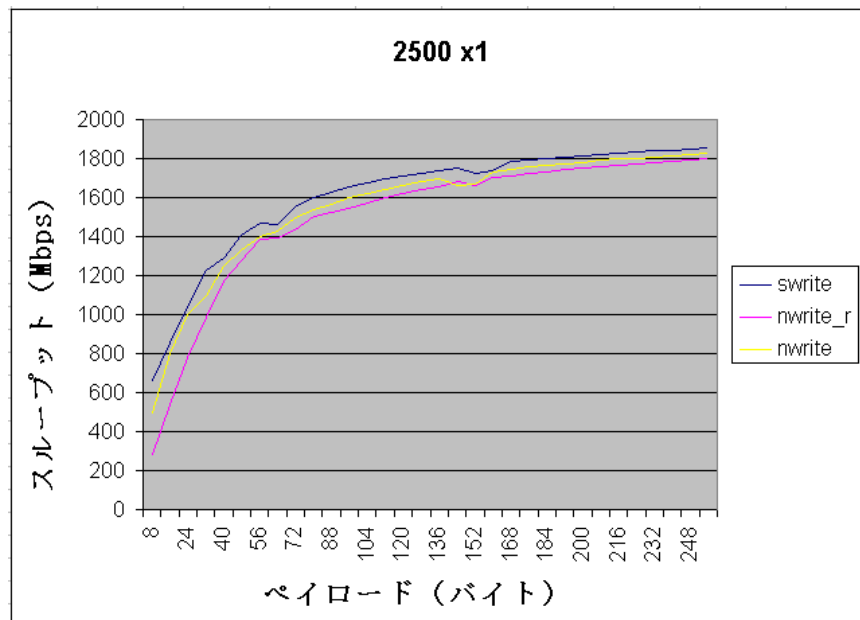
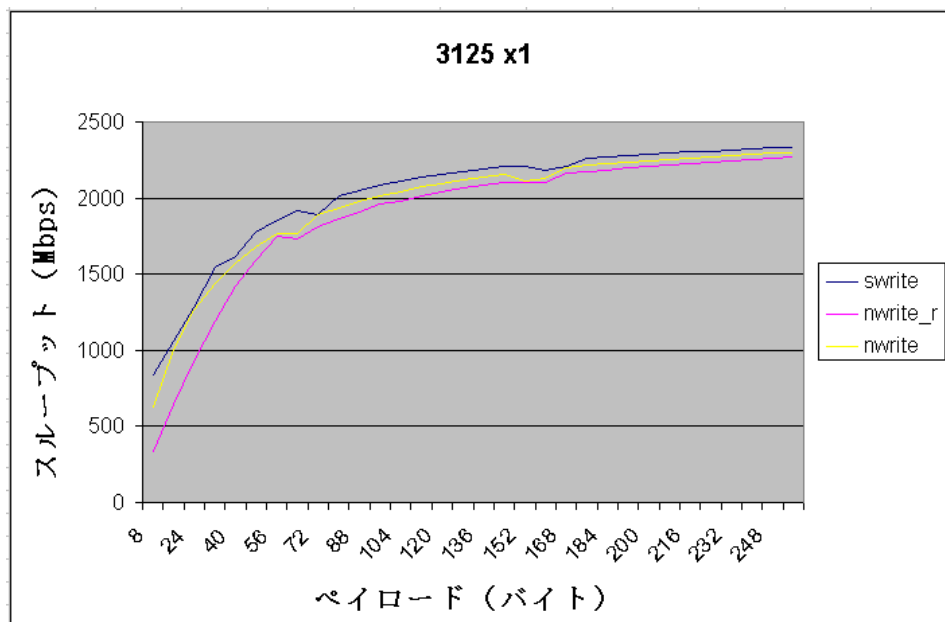


図 28. 3.125 Gbaud における x1 RapidIO MegaCore ファンクション (スループット vs. ペイロード)



デザインにおける制限

この項では、現行バージョンのリファレンス・デザインにおける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・ドライバの両方は下記のトランザクションをサポートしますが、このアプリケーションの現行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ていません。

- Maintenance Port Write
- Maintenance Port Read

デザインが DMA 手法を使用してライン・レート・トラフィックによってリンクを飽和状態にするのは、1x 1.250 Gbaud のデザイン・バリエーションのみにできます。パケット・ジェネレータ手法により、デザインは RapidIO のほかのプロトコルに対してリンクを飽和状態にすることができます。

現行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。Input/Output Slave Mapping Window 0 Control レジスタを設定すること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 と同梱されている「[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	概要
2009 年 5 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



I.S. EN ISO 9001