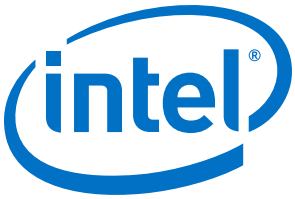


AN 784: PCI Express* リファレンス・デザインを使用した インテル® Arria® 10 デバイスのパーシャル・リコンフィグレーション

インテル® Quartus® Prime 開発デザインスイートの更新情報: **17.1**

更新情報
フィードバック

AN-784 | 2017.11.06
最新版をウェブからダウンロード: PDF | HTML

目次

1. PCI Express* リファレンス・デザインを使用した インテル® Arria® 10 デバイスのパーシャル・リコンフィグレーション

PCI Express* (PCIe*) リファレンス・デザインを使用したパーシャル・リコンフィグレーション (PR) では、インテル® Arria® 10 デバイスの PCIe リンクを介して FPGA ファブリックをリコンフィグレーションする方法を紹介します。このリファレンス・デザインは、インテル Arria 10 GX FPGA 開発ボードの Linux システム上で動作します。所定のテンプレートを使用して PR リージョンブロックを実装することで、リファレンス・デザインを要件に適応させます。PCIe を介した通信が可能な、完全に機能的なシステムでカスタム・デザインを実行してください。

パーシャル・リコンフィグレーションはフラットデザインに次の利点をもたらします。

- ランタイム・デザイン・リコンフィグレーションを可能とします。
- タイム・マルチプレクシングにより、デザインのスケーラビリティーを向上します。
- ボードを効率的に使用し、コストと消費電力を低減します。
- デザインでダイナミックなタイム・マルチプレクシングをサポートします。
- より小さなビットストリームにより、初期のプログラミング・タイムを改善します。
- ライン・アップグレードによりシステムのダウンタイムを低減します。
- リモート・ハードウェアの変更が可能なため、容易にシステム・アップグレードできます。

インテル Quartus® Prime 開発ソフトウェア・プロ・エディション、インテル Arria 10 デバイスファミリーでのパーシャル・リコンフィグレーション機能をサポートします。

関連情報

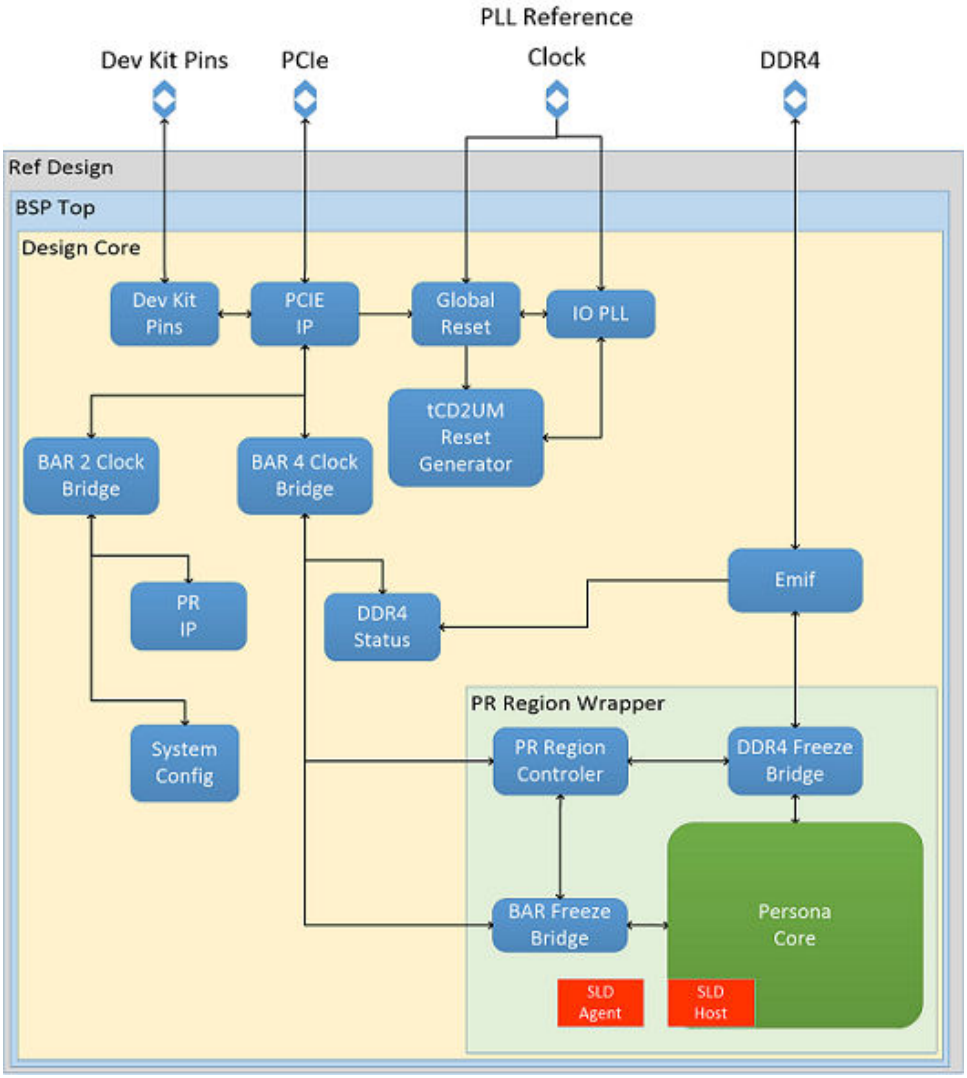
- Creating a Partial Reconfiguration Design
  パーシャル・リコンフィグレーション・デザイン・フローについての完全な情報を確認することができます。
- Partially Reconfiguring a Design on インテル Arria 10 GX FPGA Development Board
  パーシャル・リコンフィグレーション・デザインの作成についてのステップごとのチュートリアルです。
- インテル Arria 10 CvP Initialization and Partial Reconfiguration via Protocol User Guide
  CvP (Configuration via Protocol) コンフィグレーション手法についての完全な情報を確認することができます。
- インテル Arria 10 FPGA Development Kit User Guide
  インテル Arria 10 GX FPGA 開発ボードの概要とセットアップ手順についての完全な情報を確認することができます。

1.1. リファレンス・デザインの概要

リファレンス・デザインは次のコンポーネントで構成されています。

- a10_pcie_reference_design.sv—リファレンス・デザイン用のトップレベル・ラッパーです。ボード・サポート・パッケージ (BSP) サブシステムをデバイス・ピンに接続します。
- bsp_top—デザインのすべてのサブシステムを含むデザインのトップレベルです。このモジュールは、PCIe IP コア、DDR4 外部メモリー・インターフェイス IP コア、デザイン・トップ・モジュールの 3 つの主要なサブ・コンポーネントで構成されています。この抽象化のレイヤーは、シミュレーションされた Avalon-MM トランザクションを介した、デザインのトップモジュールのシミュレーションを可能にします。
- design_core—PR リージョンの生成、クロック・クロッシング Avalon-MM ロジックやパイプライン・ロジックなどのインターフェイス・コンポーネント、クロック、およびグローバルリセットを処理するデザインのコアです。

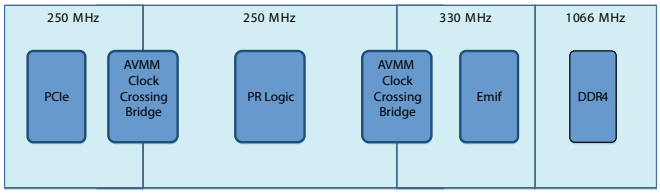
図 -1: インテル Arria 10 PCIe リファレンス・デザインのブロック図



1.1.1. クロッキング・スキーム

リファレンス・デザインは、別個の Altera IOPLL IP コア生成のクロックを作成します。このクロック生成は、250 MHz で動作する PCIe クロック・リージョンと 330 MHz で動作する EMIF クロック・リージョンから PR ロジック・クロッキングをデカップルします。タイミングを確実に収束するには、IOPLL IP コアのパラメーター化をより低いクロック周波数に修正します。

図 -2: タイミング収束



1.1.2. メモリーアドレスのマッピング

PCIe IP コアは、BAR 2 と BAR 4 の 2 つのベース・アドレス・レジスター (BAR) を介してデザインコアに接続します。BAR 2 と BAR 4 は結果として、独自の Avalon-MM インターフェイスに接続します。

BAR 4 は、Avalon-MM フリーズブリッジや PR リージョンに向けた PR リージョン・コントローラーなどのインターフェイス・コントロールに接続する他にも、最大 8 kB の PR リージョンのメモリーにも直接接続します。PR IP コアやシステム記述 ROM などのドライバーがアクセスするコンポーネントは、BAR 2 を使用します。次の表に、PCIe IP コアに向けたメモリー・アドレス・マッピングをリストします。

表 1. PCIe のメモリー・アドレス・マップ

ドメイン	アドレスマップ	Base	End
BAR 2	System Description ROM	0x0000_0000	0x0000_0FFF
BAR 2	PR IP	0x0000_1000	0x0000_103F
BAR 4	PR Region	0x0000_0000	0x0000_FFFF
BAR 4	PR Region Controller	0x0001_0000	0x0001_000F
BAR 4	DDR4 Calibration Export	0x0001_0010	0x0001_001F

リファレンス・デザインは、外部メモリー・インターフェイスの IP コアによって提供される DDR4 キャリブレーションのステータスをエクスポートします。初期化時に、EMIF IP コアは、DDR4 インターフェイスをリセットするためにトレーニングを実行します。EMIF は、キャリブレーション・フラグを使用して、このリセットの成否をレポートします。ホストは、DDR4 がこのリセット・トレーニングに成功しなかった場合に必要なアクションを決定します。よって、このインターフェイスはホストへとエクスポートされます。次の表に、外部メモリー・インターフェイスから PR リージョンへのメモリー・アドレス・マッピングをリストします。

表 2. DDR4 外部メモリー・インターフェイス (EMIF) のメモリー・アドレス・マップ

アドレスマップ	Base	End
DDR	0x0000_0000	0x7fff_ffff

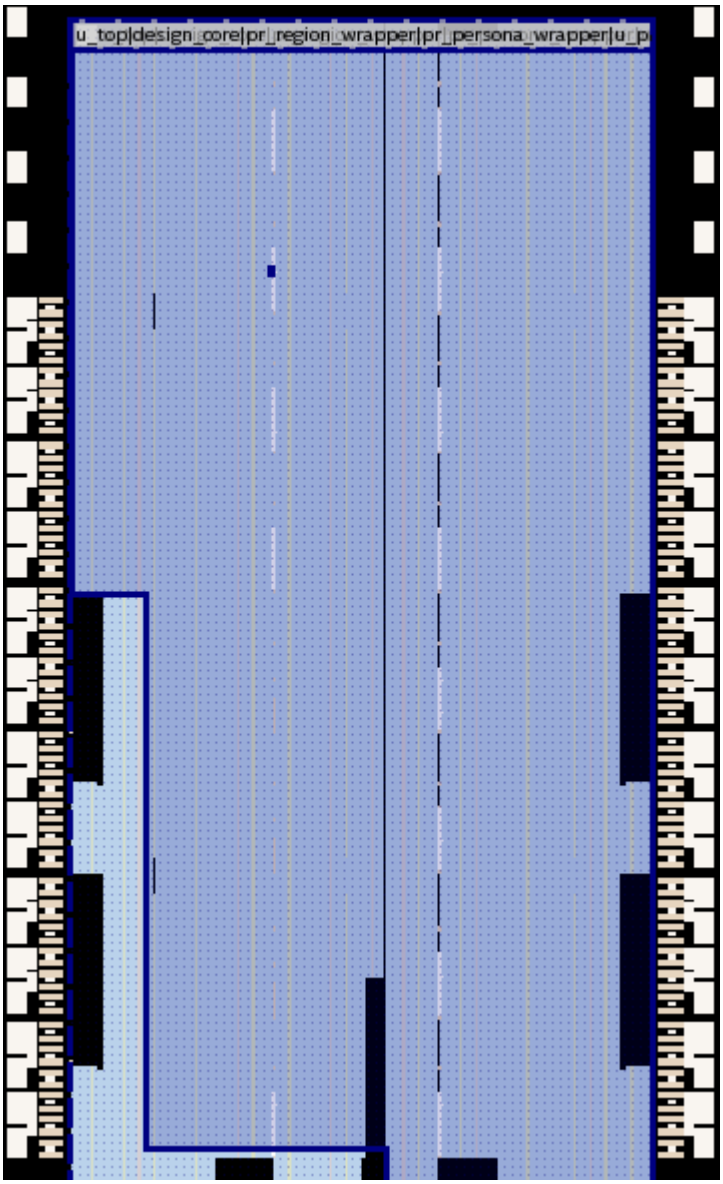
PR ロジックから Avalon-MM インターフェイスで 2 GB の DDR4 メモリースペースが使用可能です。

1.1.3. フロアプランニング

パーシャル・リコンフィグレーション・デザインのフロアプラン制約は、デバイスを物理的にパーティションします。このパーティションを実行することで、PR リージョンで使用可能なリソースが実装するすべてのペルソナで等しくなります。

PR リージョンで使用可能なファブリックの量を最大にするには、リファレンス・デザインでスタティック・リージョンを可能な限り小さなリージョンに制限します。

図 -3: リファレンス・デザインのフロアプラン



フロアプランニングについての詳細は、*インテル Quartus Prime 開発ソフトウェア・プロ・エディション Handbook* の Volume 1*Floorplan the Partial Reconfiguration Design* の項を参照してください。

関連情報

Floorplan the Partial Reconfiguration Design

1.2. 使用開始に際して

この項では、リファレンス・デザインを実行するにあたっての要件と手順について説明します。

1.2.1. ハードウェアおよびソフトウェア要件

リファレンス・デザインを使用するには、次のハードウェア・ツールとソフトウェア・ツールが必要です。

- DDR4 モジュールが Hi-Lo インターフェイスに接続された インテル Arria 10 GX FPGA 開発ボード
- カーネルバージョン 3.10 以降の Linux オペレーティング・システム
- ホストマシーンへのスーパー・ユーザー・アクセス
- インテル Arria 10 GX FPGA 開発ボードを差し込むための PCIe スロット
- PCIe リファレンス・デザインを使用したこの PR に対するオープン・ソース・ドライバー
- インテル Quartus Prime 開発ソフトウェア・プロ・エディション v.17.1
- インテル FPGA ダウンロード・ケーブルドライバー

注意: PCIe リファレンス・デザインを使用したこの PR に向けた開発されたオープン・ソース・ドライバーは、CentOS 7 を使用して検証されています。

1.2.2. インテル Arria 10 GX FPGA 開発キットのインストール

Linux システムでの インテル Arria 10 GX FPGA 開発ボードのインストールと電源投入についての詳細な手順は、*インテル Arria 10 FPGA Development Kit User Guide* を参照してください。

注意: ボードに電源を投入する前に、DIP スイッチバンク (SW6) のスイッチ 4 (FACT0RY) を **ON** に設定してください。このスイッチを **ON** に設定すると、ブート時にフラッシュメモリーのファクトリー・イメージ・エリアがロードされます。リファレンス・デザインをこのファクトリー・イメージ・エリアにプログラミングします。リファレンス・デザインをボードにフラッシュする手順の詳細については、*Bringing Up the Reference Design* を参照してください。

関連情報

- インテル Arria 10 FPGA Development Kit User Guide
- リファレンス・デザインの立ち上げ (8 ページ)

1.2.3. Linux ドライバーのインストール

リファレンス・デザインには、リファレンス・デザインに向けて開発・検証されたオープンソースの Linux ドライバー用の完全なソースコードが含まれています。

このデザインに向けた Linux ドライバーには、debugfs をマウントする必要があります。debugfs がマウントされているかどうかを確認するには、次のコマンドを実行します。

```
mount | grep ^debugfs
```

debugfs ファイルシステムについての詳細は、CentOS の資料を参照してください。

重要: CentOS 7 は、このドライバーがサポートしている唯一の OS です。

このドライバーをインストールする前に、必要なパッケージをすべてインストールしなければいけません。必要なパッケージをインストールするには、次のコマンドを実行します。

```
yum groupinstall “Development Tools”
```

```
yum install ncurses-devel
```

```
yum install qt-devel
```

```
yum install hmaccalc zlib-devel binutils-devel elfutils-libelf-devel
```

ドライバーをインストールするには、次の手順に従ってください。

1. ドライバーのソースコードがマシーンにコピーされていることを確認します。このソースコードは、次の箇所にあります。

   https://github.com/intel/fpga-partial-reconfig/tree/master/software/drivers

2. 必要なドライバーモジュールをすべてコンパイルするには、次のコマンドを実行します。

   ```
   make
   ```

   このコマンド実行後、次の 3 つのカーネル・オブジェクト・ファイルがドライバーのソース・ディレクトリーに存在することを確認してください。

   - `fpga-mgr-mod.ko`
   - `fpga-pcie-mod.ko`

3. 適切な箇所にモジュールをコピーし、モジュール依存データベースを更新するには、次のコマンドを実行します。

   ```
   sudo make install
   ```

4. ドライバーのインスタンスを各 インテル FPGA デバイスに分配するには、次のコマンドを実行します。

   ```
   sudo modprobe fpga-pcie-mod
   ```

5. ドライバーのインストールが成功したかどうかを確認するには、次のコマンドを実行します。

   ```
   lspci –vvvd1172:
   ```

   問題なくインストールが完了すると、最後に次の内容が表示されます。

   ```
   Kernel driver in use: fpga-pcie
   Kernel modules: fpga_pcie_mod
   ```

   注意: 上記のコマンドは、デザインがボードにロード済みで、かつコンピューターが再起動されている場合にのみ実行されます。

1.2.4. リファレンス・デザインの立ち上げ

リファレンス・デザインは次のウェブサイトで利用可能です。

https://github.com/intel/fpga-partial-reconfig

リファレンス・デザインにアクセスするには、`ref_designs` サブフォルダーに移動します。`a10_pcie_devkit_cvp` フォルダーを Linux システムのホーム・ディレクトリーにコピーします。

ボードでリファレンス・デザインを立ち上げるには、次の手順に従ってください。

1. ホストマシーンで利用可能な PCIe スロットに インテル Arria 10 GX FPGA 開発ボードを差し込みます。
2. ホストマシーンの ATX 補助電源コネクターを開発ボードの 12 V ATX 入力 J4 に接続します。
3. ホストマシーンに電源を投入します。
4. FPGA 開発ボードへの micro-USB ケーブルの接続を確認します。JTAG チェーンを実行する他のアプリケーションが使用されていないことを確認してください。
5. システムの `a10_pcie_devkit_cvp/software/installation` フォルダーに移動します。
6. ボードの既存のファクトリー・イメージをリファレンス・デザインで上書きする場合、`flash.pl` スクリプトを実行します。
7. 接続した JTAG ケーブルのナンバー (例、`perl flash.pl 1`) を引数としてスクリプトへ渡します。

   このスクリプトを実行すると、`flash.pof` ファイルの内容でデバイスがコンフィグレーションされます。このパラレル・オブジェクト・ファイルは、`output_files` ディレクトリーに存在する `a10_pcie_devkit_cvp.sof` ファイルに直接由来します。`flash.pof` ファイルは、リファレンス・デザインへのベースイメージとして動作します。

   注意: このスクリプトを実行する前に、デザインのコンパイルに問題がないことを確認してください。
8. ホストマシーンを再起動します。

1.3. リファレンス・デザインのコンポーネント

リファレンス・デザインには次のコンポーネントが含まれています。

1.3.1. BSP Top

この Platform Designer システムには、このリファレンス・デザインのすべてのサブシステムが含まれています。このシステムは、トップレベル・デザイン、PCIe IP コア、および DDR4 外部メモリー・インターフェイス IP コアの 3 つの主要コンポーネントから構成されています。このシステムは、`a10_pcie_ref_design.sv` ラッパーを介して外部ピンに接続します。

1.3.1.1. PCI Express IP コア

PCI Express IP コア用の インテル Arria 10 ハード IP は Gen3x8 で、256 ビットのインターフェイスを備えており、250MHz で動作します。

次の表は、リファレンス・デザインが使用するデフォルトの設定とは異なる PCI Express IP コアのコンフィグレーション・フィールドの情報を提供します。

表 3. PCI Express IP コアのコンフィグレーション

設定	パラメーター	値
System Settings	**Application interface type**	DMA を備えた Avalon-MM
	Hard IP mode	Gen3:x8、インターフェイス: 256 ビット、250 MHz
	Port type	ネイティブ・エンドポイント

continued...

設定	パラメーター	値
	RX buffer credit allocation for received requests vs completions	Low
Avalon-MM Settings	**Enable control register access (CRA) Avalon-MM slave port**	ディセーブル
Base Address Registers - BAR2	**Type**	32 ビットのプリフェッチ不可能なメモリー
Base Address Registers - BAR4	**Type**	32 ビットのプリフェッチ不可能なメモリー
Device Identification Registers	**Vendor ID**	0x00001172
	Device ID	0x00005052
	Revision ID	0x00000001
	Class code	0x00ea0001
	Subsystem Vendor ID	0x00001172
	Subsystem Device ID	0x00000001
PCI Express/PCI Capabilities - Device	**Maximum payload size**	256 バイト
Configuration, Debug, and Extension Options	**Enable Arria 10 GX FPGA Development Kit Connection**	Enable
PHY Characteristics	**Requested equalization far-end TX preset**	Preset 9

注意: PCI Express IP コアを Platform Designer システムの一部として初期化します。

1.3.1.2. インテル Arria 10 DDR4 外部メモリー・インターフェイスの IP コア

`ddr4_emif` ロジックには、インテル Arria 10 外部メモリー・インターフェイスの IP コアが含まれています。この IP コアは、1066.0 MHz で動作する 64 ビットのインターフェイスを備えた DDR4 外部メモリーとインターフェイス接続します。また、IP コアは 2 GB の DDR4 SDRAM メモリースペースも提供します。EMIF の Avalon-MM スレーブは 300 MHz クロックで動作します。

次の表に、DDR4 HILO を備えた インテル Arria 10 GX FPGA 開発キットのプリセットとは異なる インテル Arria 10 外部メモリー・インターフェイス IP コアのコンフィグレーション・フィールドをリストします。

表 4. インテル Arria 10 DDR4 外部メモリー・インターフェイスの IP コンフィグレーション

設定	パラメーター	値
Memory - Topology	**DQ width**	64
	DQ pins per DQS group	8
	Number of DQS groups	8
	Alert# pin placement	Address/Command ピンを備えた I/O レーン
	Address/Command I/O lane of ALERT#	3
	Pin index of ALERT#	0

1.3.1.3. デザイントップ

このコンポーネントはデザインの中核を形成し、次のものを含みます。

- リセットロジック
- PR リージョン
- パーシャル・リコンフィグレーション IP コア
- Avalon MM トランザクションに向けたクロック・クロッシングおよびパイプライニング
- システム記述 ROM
- PLL

1.3.1.3.1. グローバル・リセット・ロジック

PLL は、このデザインのメインクロックを生成します。`pcie ip`、`pr ip`、および `ddr4 emif` を除くクロックはすべて、この 250 MHz のクロックを使用して動作します。 PCIe は、PLL リセット信号とグローバルリセット信号を生成します。電源投入時にカウントダウン・タイマーである `tcd2um` は、内部の 50 MHz オシレーターを使用して、遅延が 830 µs となるまでカウントダウンします。タイマーがこの遅延に到達するまで、PLL はリセット状態で保持され、ロックされた信号をディアサートします。この動作はデザインをフリーズします。また、PLL がロックしている信号は、 PCIe リセットにより OR 演算されるため、デザインもリセット状態で保持されます。タイマーが 830µs に到達すると、デザインは通常の状態で動作し、既知の状態になります。

1.3.1.3.2. PR リージョンラッパー

PR リージョンラッパーには、PR リージョン・コントローラー、フリーズブリッジ、およびペルソナが含まれています。このリージョン・コントローラーは、PR を初期化する目的で Avalon-MM インターフェイスを介してドライバーと相互作用し、フリーズおよびスタート・リクエストの開始に際し、PR リージョンと通信するためのブリッジとして機能します。

1.3.1.3.3. インテル Arria 10 パーシャル・リコンフィグレーション・コントローラー IP コア

インテル Arria 10 パーシャル・リコンフィグレーション・コントローラー IP コアを使用して、PR リージョンへのフリーズリクエストを開始します。PR リージョンは、フリーズリクエストの確認応答ですべての動作を終了させます。フリーズブリッジは、PR リージョンへの Avalon-MM インターフェイスを停止し、パーシャル・リコンフィグレーション中に PR リージョンに向けて作成されたトランザクションに対し適切に応答します。PR 完了時に、リージョン・コントローラーは停止リクエストを発行し、リージョンが確認応答し、適切に動作することを可能にします。このリファレンス・デザインで提供される `fpga-region-controller` プログラムはこれらの機能を実行します。

リファレンス・デザインは、パーシャル・リコンフィグレーション・リージョン・コントローラー IP コアを内部ホストとして動作するようにコンフィグレーションします。デザインは、Avalon-MM インターフェイスのインスタンスを介してこの IP コアを PCI Express IP コアに接続します。この PR IP コアのクロック対データ比は 1 です。よって、PR IP コアは、暗号化または圧縮された PR データを処理できません。

次の表に、プリセット設定とは異なる PCI Express IP コア用の インテル Arria 10 ハード IP のコンフィグレーション・フィールドをリストします。

パラメーター	値
Enable JTAG debug mode	Disable
Enable Avalon-MM slave interface	Enable
Input data width	32

1.3.1.3.4. パーシャル・リコンフィグレーション・ロジック

リファレンス・デザインは次のペルソナを提供します。

表 5. リファレンス・デザインのペルソナ

ペルソナ	説明
DDR4 access	メモリースパン全体にわたってスウィープを実行し、最初にライトを行い、次に各アドレスをリードします。.
Basic DSP	27x27 DSP 乗算器へのアクセスを提供し、ハードウェア・アクセラレーションを示します。
Basic arithmetic	基本的な 32 ビットの符号なし加算器を含み、ハードウェア・アクセラレーションを示します。
Game of Life	8x8 のライフゲームを含み、ハードウェア・アクセラレーションを示します。

各ペルソナは、固有の識別番号を表すために `pr_data` レジスターに 8 ビットの `persona_id` フィールドを備えています。1 個の 32 ビット制御レジスターと 16 個の I/O レジスターが、8 ビットの `persona_id` に続きます。16 個の I/O レジスターはそれぞれが 32 ビットで、デバイス入力用に 8 ビット、デバイス出力用に 8 ビットを備えています。各ペルソナはこれらのレジスターを別々の方法で使用します。詳細は、各ペルソナのソースコードを参照してください。

さらに、リファレンス・デザインはカスタムのペルソナを実装するためのテンプレートを提供します。このテンプレートのペルソナを使用すると、RTL の変更、レジスターファイルとインターフェイスするラッパーの作成、、コンパイル、デザインの実行が可能となります。

1.4. リファレンス・デザインのコンパイル

インテル Quartus Prime は、PR ペルソナをコンパイル、そして実行するフロースクリプトを提供します。partial reconfiguration フロースクリプトを作成するには、次のコマンドを生成し、実行します。

1. インテル Quartus Prime のコマンドシェルより、次のコマンドを実行してフロー・テンプレートを作成します。

   ```
   quartus_sh --write_flow_template -flow a10_partial_reconfig
   ```

   インテル Quartus Prime は、`a10_partial_reconfig/flow.tcl` ファイルを生成します。
2. 生成した `a10_partial_reconfig/setup.tcl.example` の名称を `a10_partial_reconfig/setup.tcl` へと変更し、パーシャル・リコンフィグレーション・プロジェクトの詳細を指定するようにスクリプトを修正します。
   a. プロジェクトの名称を定義するには、次の行を更新します。

      ```
      define_project a10_pcie_devkit_cvp
      ```

   b. ベースバージョンを定義するには、次の行を更新します。

      ```
      define_base_revision a10_pcie_devkit_cvp
      ```
3. PR パーティションの名称とリビジョンを実装する合成リビジョンと一緒に各パーシャル・リコンフィグレーション実装のリビジョンを定義するには、次の行を更新します。

   ```
   define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_ddr4_access\
   -partition_name pr_partition\
   -source_rev_name synth_ddr4_access
   ```

```
define_pr_impl_partition -impl_rev_name
a10_pcie_devkit_cvp_basic_arithmetic \ -partition_name pr_partition \ -
source_rev_name  synth_basic_arithmetic

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_dsp \ -
partition_name pr_partition \ -source_rev_name  synth_basic_dsp

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_gol \ -
partition_name pr_partition \ -source_rev_name  synth_gol
```

注意: すべてのリビジョン・プロジェクトは、`a10_pcie_devkit_cvp.qpf` と同じディレクトリー内にしなければいけません。そうでなければ、フロースクリプトは状況に応じて適切に更新します。

4. **Tools** > **Tcl Scripts** の順でクリックします。**Tcl Scripts** ダイアログボックスが表示されます。
5. **Add to Project** をクリックし、`a10_partial_reconfig/flow.tcl` を選択・展開します。
6. Libraries ペインの `a10_partial_reconfig/flow.tcl` を選択し、**Run** をクリックします。

   このスクリプトは、4 つのペルソナに対して合成を実行します。インテル Quartus Prime は、各ペルソナに対し SRAM オブジェクト・ファイル (`.sof`)、Partial-Masked SRAM オブジェクト・ファイル (`.pmsf`)、ロウ・バイナリ・ファイル (`.rbf`) を生成します。

   インテル Quartus Prime コマンドシェルからこのスクリプトを実行するには、次のコマンドを入力します。

```
quartus_sh -t a10_partial_reconfig/flow.tcl -setup_script
a10_partial_reconfig/setup.tcl
```

1.5. リファレンス・デザインの検証

リファレンス・デザインは、FPGA ボードをプログラミングする目的で次のユーティリティーを提供しています。

- `program-fpga-jtag`
- `program_fpga_pcie`
- `fpga-region-controller`

このデザインには、デバイスと通信し、各ペルソナを示す `example_host_uio` アプリケーションも含まれています。

1.5.1. program_fpga_jtag

`program_fpga_jtag` スクリプトを使用すれば、再起動をしなくてもデバイス全体 (フルチップ・プログラミング) することが可能です。

`program_over_jtag` は、次の内容を実行します。

- デバイスのプログラミングに インテル Quartus Prime Programmer を使用します。
- SRAM オブジェクト・ファイル (.sof) を受け入れ、JTAG インターフェイスを介してターゲットデバイスをコンフィグレーションします。
- ドライバーと通信して、次の内容を実行します。
  - アップストリーム advanced error reporting (AER) をディセーブルします。
  - ステートを保存します。
  - AER を元の状態に戻します。
  - ステートを元の状態に戻します。

表 6. program_fpga_jtag コマンドラインのオプション

オプション	説明
-f=, --file=[<*filename*>]	.sof ファイルの名称を指定します。
-c= ,--cable=[<*cable number*>]	プログラマー・ケーブルを指定します。
-d=, --device=[<*device index*>]	PCIe デバイスの番号を指定します。例、-d=*0000:03:00.0*
-h, --help	program_fpga_jtag スクリプトへのヘルプを提供します。

注意: デバイス・インデックスを取得するには、次のコマンドを使用します。

```
/sbin/lspci -d1172:
```

例えば、コマンドが次のような出力を返す場合、

```
03:00.0 Class ea00: Altera Corporation Device 5052 (rev 01)
```

最初の値がデバイス・インデックスです。この値に、0000 を追加します。この場合のデバイス・インデックスは、*0000.03:00.0* となります。

1.5.2. program_fpga_pcie

パーシャル・リコンフィグレーションを実行するには、program_fpga_pcie ユーティリティーを使用します。このスクリプトは所定のペルソナの .rbf ファイルを受け入れます。このスクリプトは次の内容を実行します。

- デバイスのサブドライバーが存在する場合、それらを削除するためにドライバーと通信します。
- フリーズをアサート/ディアサートするために fpga-region-controller スクリプトと通信します。
- .rbf をパーシャル・リコンフィグレーション・コントローラー IP コアにライトします。
- PR が成功した場合に必要となるサブドライバーが存在する場合、それらを再配置します。

表 7. program_fpga_pcie コマンドラインのオプション

オプション	説明
-f=, --file=[<*filename*>]	.rbf ファイルの名称を指定します。
-d=, --device=[<*device index*>]	PCIe デバイスの番号を指定します。例、-d *0000:03:00.0*
-r=, --region=[<*region offset*>]	ブリッジイネーブルのフリーズに使用するターゲット FPGA リージョン・コントローラー・オフセットを指定します。このオプションは、次の内容を実行します。

continued...

オプション	説明
	• フリーズ・リクエストの交換を開始します。 • パーシャル・リコンフィグレーション中およびパーシャル・リコンフィグレーション後に PR リージョンに向けてリセットをアサート/ディアサートします。 例: `program-fpga-pcie --file=<region.rbf> --device=0000:03:00.0 --region=10000` *注意:* リージョンオフセットの値は、16 進数でなければいけません。 PR リージョンは、この動作中にリセットされます。
`-h, --help`	`program_jtag_pcie` スクリプトのヘルプを提供します。

1.5.3. example_host_uio

`example_host_uio` モジュールは、FPGA デバイスアクセスを示します。このアプリケーションは、各ペルソナと相互作用し、ペルソナの内容と機能を検証します。

このプログラムは、PCIe デバイスの番号を必要とし、テストデータを生成するシード、実行された検証の回数、追加の情報を表示する冗長などのオプションのパラメーターが続きます。

表 8. example_host_uio コマンドラインのオプション

オプション	説明
`-s=, --seed [<seed>]`	番号の生成に使用するシードを指定します。デフォルトの値は、1 です。
`-d, --device [<device index>]`	PCIe デバイスの番号を指定します。例、`-d 0000:03:00.0`
`-v, --verbose`	実行中に別の情報のプリントが可能です。
`-n, --iterations`	実行する検証のイタレーションを指定することが可能です。
`-h, --help`	`example_host` アプリケーションのヘルプを提供します。

Signal Tap

リファレンス・デザインは、階層ハブを介した PR リージョンのシグナルタップをサポートします。この機能により、特定のペルソナのオンチップ・デバッグが容易に実行できます。このスタティック・リージョンには SLD エージェントが含まれますが、これは SLD ホストと通信します。シグナルタップする予定のペルソナの SLD ホストは、初期化する必要があります。また、所定のペルソナの合成のみのリビジョンに `.stp` ファイルを含める必要があります。`.stp` ファイルが特定のペルソナに対して固有のものであれば、ベースリビジョン、つまり他のペルソナの `.qsf` ファイルにシグナルタップ・ファイルを含めないでください。

PR ロジックをシグナルタップする場合は、次のガイドラインに従ってください。

- 合成前のレジスターを使用します。
- インスタンスはパーティション境界を超えないようにします。
- パーティション境界にまたがってトリガーしないようにします。

階層ハブを使用したインフラストラクチャーについての詳細は、インテル Quartus Prime 開発ソフトウェア・プロ・エディションハンドブック Volume 3 の *Debugging Partial Reconfiguration Designs Using Signal Tap Logic Analyzer* の項を参照してください。

関連情報

Signal Tap Logic Analyzer を使用したパーシャル・リコンフィグレーションのデバッグ

1.5.3.1. アプリケーション例のコンパイル

リファレンス・デザインのソフトウェア・アプリケーションは、`software/util` ディレクトリー内で利用可能です。各アプリケーションには、付随する `Makefile` を持つサブ・ディレクトリー・ストラクチャーが含まれています。

アプリケーション例をビルドするには、次の内容を実行します。

1. `example_host` モジュールをコンパイルするには、Linux シェルから次の内容を入力します。

```
cd source/util
make
```

このコマンドはサブディレクトリー内に次のような実行ファイルを生成します。

```
./example_host_uio -d 0000:03:00.0 -s 1  -n 100 -v
```

このコマンドであれば、デバイス `0000:03:00.0` を使用、値が 1 の入力生成をシード、100 回のイタレーションを実行、現在の状態でより多くの情報をプリントします。

1.5.3.2. アプリケーション例を使用したデザインのプログラミング

デバイスが `0000:03:00.0` でケーブル番号が 1 であると仮定します。

次の手順は、提供されたスクリプトを使用したデザインのプログラミングを表しています。

1. デザインのコンパイル後に DDR access ペルソナの `.sof` ファイルをプログラミングするには、Linux シェルから次の内容を入力します。

```
program-fpga-jtag -f=a10_pcie_devkit_cvp_ddr4_access.sof -c=1 -
d=0000:03:00.0
```

2. デザインの機能性を検証するには、Linux シェルから次の内容を入力します。

```
source/util/example_host/example_host_uio -d 0000:03:00.0
```

3. デザインのコンパイル後にレジスターファイル・ペルソナの `.rbf` ファイルをプログラミングするには、Linux シェルから次の内容を入力します。

```
program-fpga-pcie -f=a10_pcie_devkit_cvp_basic_dsp.pr_partition.rbf -
d=0000:03: 00.0 -r=10000
```

4. デザインの機能性を検証するには、Linux シェルから次の内容を入力します。

```
software/util/example_host/example_host_uio -d 0000:03:00.0
```

5. デザインのコンパイル後に Basic arithmetic ペルソナの `.rbf` ファイルをプログラミングするには、Linux シェルから次の内容を入力します。

```
program-fpga-pcie -f=a10_pcie_devkit_cvp_basic_arithmetic.pr_partition.rbf
-
d=0000:03:00.0 -r=10000
```

6. デザインの機能性を検証するには、Linux シェルから次の内容を入力します。

```
software/util/example_host/example_host_uio 0000:03:00.0
```

1.6. カスタムのペルソナを使用したリファレンス・デザインの拡張

このリファレンス・デザインは PCIe を介した PR に向けて独自のペルソナを作成するテンプレート例を提供します。独自のペルソナを使用してリファレンス・デザインを拡張するには次を実行します。

1. `a10_pcie_devkit_cvp` フォルダーに移動します。

```
cd a10_pcie_devkit_cvp
```

2. `pr_logic_impl_template.qsf.template` 実装リビジョンファイルと `pr_logic_synth_template.qsf.template` 合成ファイルのコピーを作成します。

```
cp pr_logic_impl_template.qsf.template <persona_impl_revision_name>.qsf
cp pr_logic_synth_template.qsf.template <persona_synth_revision_name>.qsf
```

3. フォルダーを作成し、そのフォルダーにペルソナ固有の RTL をコピーします。

```
mkdir <persona_name>
cp <custom_persona>.sv <persona_name>/
```

4. カスタムのトップレベル・エンティティーは、`source/templates/pr_logic_template.sv` ファイルで定義される `custom_persona` モジュールのポートに一致しなければいけません。次の例は、PCIe レジスターファイルを介して制御される Avalon-MM インターフェイスを備えたデザインとのインターフェイス接続を示しています。

```
module custom_persona #(
parameter REG_FILE_IO_SIZE = 8
)(
//clock
input wire clk,

//active low reset, defined by hardware
input wire rst_n,
//Persona identification register, used by host in host program
output wire [31:0] persona_id,
//Host control register, used for control signals.
input wire [31:0] host_cntrl_register,
// 8 registers for host -> PR logic communication
input wire [31:0] host_pr [0:REG_FILE_IO_SIZE-1],
// 8 Registers for PR logic -> host communication
output wire [31:0] pr_host [0:REG_FILE_IO_SIZE-1]
);
```

カスタマイズに向けて任意のパラレル I/O ポート (PIO) レジスターを使用します。`host_pr` レジスターはペルソナからのデータをホストマシーンに送信します。`pr_host` レジスターはホストマシーンからのデータをペルソナに送信します。

5. トップレベル・エンティティー・ファイル内で、ペルソナ ID を任意の 32 ビットの値に指定します。

```
assign persona_id  = 32'h0000_aeed;
```

注意: テンプレート例では、8 ビットだけを使用していますが、32 ビットまででであれば任意の値を指定できます。

6. `pr_host` レジスターの未使用出力ポートを 0 に設定します。

```
generate
genvar i;
//Tying unused output ports to zero.
for (i = 2; i < REG_FILE_IO_SIZE; i = i + 1) begin
 assign pr_host [i] = 32'b0;
end
endgenerate
```

7. 次のアサインメントを含むように persona_synth_revision_name.qsf を変更します。

```
set_global_assignment -name TOP_LEVEL_ENTITY <custom_persona>
set_global_assignment -name SYSTEMVERILOG_FILE <persona_name>/
<custom_persona>.sv
set_global_assignment -name QSYS_FILE <persona_specific_qsys_file>
set_global_assignment -name IP_FILE <persona_specific_ip_file>
```

8. 新しい PR 実装を定義するために、パーシャル・リコンフィグレーション・フローのスクリプトを更新します。
9. 合成リビジョンと実装リビジョンを含めるように a10_pcie_devkit_cvp.qpf プロジェクトファイルを更新します。

```
PROJECT_REVISION = "<persona_synth_revision_name>"
PROJECT_REVISION = "<persona_impl_revision_name>」
```

10. リビジョンをコンパイルします。

    PR デザインにカスタムのペルソナを追加する場合の詳細については、アプリケーション・ノート *Partially Reconfiguring a Design on インテル Arria 10 GX FPGA Development Board* の *Adding a New Persona to the Design* の項を参照してください。

関連情報

Adding a New Persona to the Design

1.7. 改訂履歴

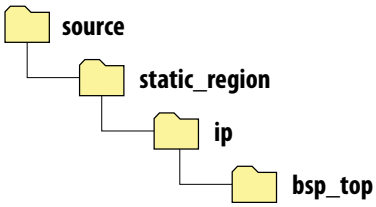
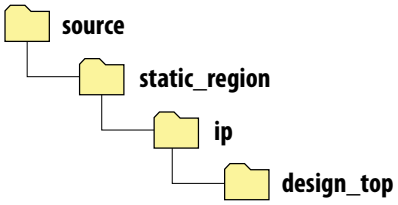
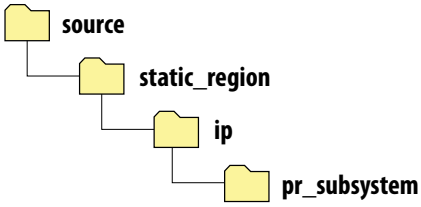
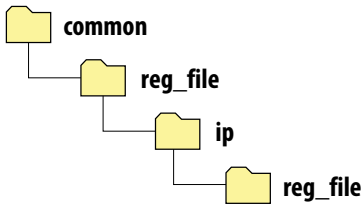
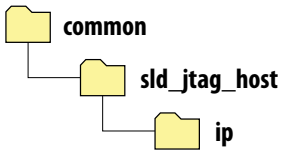
次の表は本資料の改訂履歴です。

表 9. 改訂履歴

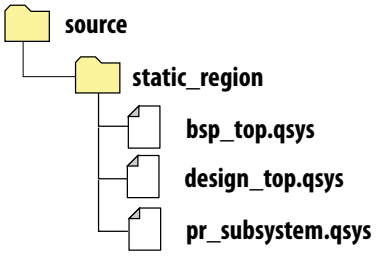
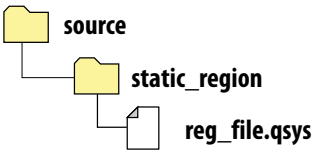
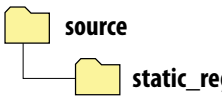
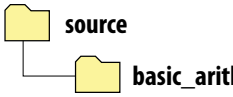
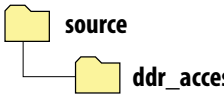
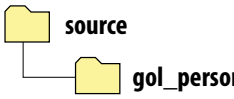
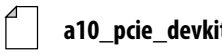
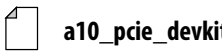
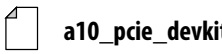
ドキュメント・バージョン	ソフトウェア・バージョン	変更内容
2017.11.06	17.1.0	• リファレンス・デザインの要件を追加し、*ハードウェアおよびソフトウェアの要件*の項を更新しました。 • オープンソース Linux ドライバーのインストールについての詳細な手順を追加し、*Linux ドライバーのインストール*の項を更新しました。 • バージョンを更新しました。 • 編集上の軽微な変更を行いました。
2017.05.08	17.0.0	• リファレンス・デザインのフローを更新しました。 • 新しいペルソナであるライフゲームを追加しました。
2016.10.31	16.1.0	初版

A. リファレンス・デザイン・ファイル

表 10. リファレンス・デザイン・ファイルのリスト

種類	ファイル/フォルダー	説明
IP ファイル		インテル Arria 10 外部メモリー・インターフェイス IP コア、PCI Express IP コア用の インテル Arria 10 ハード IP、devkit ピンに向けた IP ファイルが含まれます。
		インテル Arria 10 リコンフィグレーション・コントローラー IP コア、システム記述 ROM、キャリブレーション I/O、およびすべてのインターフェイス・コンポーネントに向けた IP ファイルが含まれます。
		フリーズブリッジ、リージョン・コントローラー、および JTAG SLD エージェントが含まれます。
		全ペルソナで共通のレジスター・ファイル・システムに向けたすべての IP ファイルが含まれます。
		PR リージョンのシグナルタッピング用の JTAG SLD ホストが含まれます。これらのファイルはすべてのペルソナに適用可能です。

continued...

種類	ファイル/フォルダー	説明
Platform Designer システムファイル		次の 3 つの Platform Designer (Standard)サブシステムが含まれます。 • bsp_top.qsys はトップレベルのサブシステムです。PCIe IP コアと外部メモリー・インターフェイス IP コアが含まれます。 • design_top.qsys はスタティック・リージョンです。Avalon-MM インターフェイス・ロジック、リセットロジック、および PR リージョン・コントローラー IP コアを含んでいます。 • pr_subsystem.qsys は、PR リージョンとの通信・相互作用に必要なすべてのロジックが含まれます。
		各ペルソナが使用する単純なレジスターファイルが含まれます。このファイルには、ID、コントロール、8 入力および 8 出力レジスターといった 32 ビットレジスターを含んでいます。
SystemVerilog デザインファイル		トップレベルのラッパーを含んでいます。また、3 つのサブシステムと PR リージョンラッパーの汎用コンポーネントに向けた SystemVerilog の記述も含まれています。
		Basic arithmetic ペルソナ用のすべてのソースファイルが含まれています。
		DDR4 access ペルソナ用のすべてのソースファイルが含まれています。
		ライフゲーム・ペルソナ用のすべてのソースファイルが含まれています。
		ペルソナ・コンフィグレーション用のテンプレートを使用するペルソナ例です。これらの例は、カスタムのペルソナ RTL をリファレンス・デザインへ統合する方法を示します。
メモリーファイル		システム記述 ROM に使用されます。
Synopsys デザイン制約ファイルです。		デザインの合成制約です。
		例外を提供します。
		pcie_subsystem_alt_pr.ip ファイルから制約を自動生成します。
インテル Quartus Prime プロジェクト・ファイル		すべてのリビジョンが含まれます。
インテル Quartus Prime 設定ファイル		DDR4 access ペルソナ用のベースリビジョン設定ファイルです。

continued...

種類	ファイル / フォルダー	説明
	synth_ddr4_access.qsf	DDR4 access ペルソナ用の合成リビジョン設定ファイルです。
	synth_basic_dsp.qsf	Basic DSP ペルソナ用の合成リビジョン設定ファイルです。
	synth_basic_arithmetic.qsf	Basic arithmetic ペルソナ用の合成リビジョン設定ファイルです。
	synth_gol.qsf	ライフゲーム・ペルソナ用の合成リビジョン設定ファイルです。
	a10_pcie_devkit_cvp_ddr4_access.qsf	DDR4 access ペルソナ用の実装リビジョン設定ファイルです。
	a10_pcie_devkit_cvp_basic_dsp.qsf	Basic DSP ペルソナ用の実装リビジョン設定ファイルです。
	a10_pcie_devkit_cvp_basic_arithmetic.qsf	Basic arithmetic ペルソナ用の実装リビジョン設定ファイルです。
	a10_pcie_devkit_cvp_gol.qsf	ライフゲーム・ペルソナ用の実装リビジョン設定ファイルです。
PR 設定スクリプト	**setup.tcl**	PR フローのコンパイルを指定します。