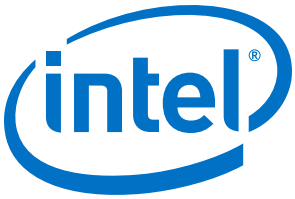


AN 813: PCI Express* リファレンス・デザインを使用した Arria® 10 デバイスの階層的なパーシャル・リコンフィグレーション

インテル® Quartus® Prime 開発デザインスイートの更新情報: **18.1**

更新情報
フィードバック

AN-813 | 2018.09.24
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 Arria® 10 デバイスの階層的なパーシャル・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階層的なパーシャル・リコンフィグレーション (HPR) では、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、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 プロ・エディション開発ソフトウェアの v.18.1 では、パーシャル・リコンフィグレーションに向けて新しく簡略化されたコンパイルフローを導入しています。

関連情報

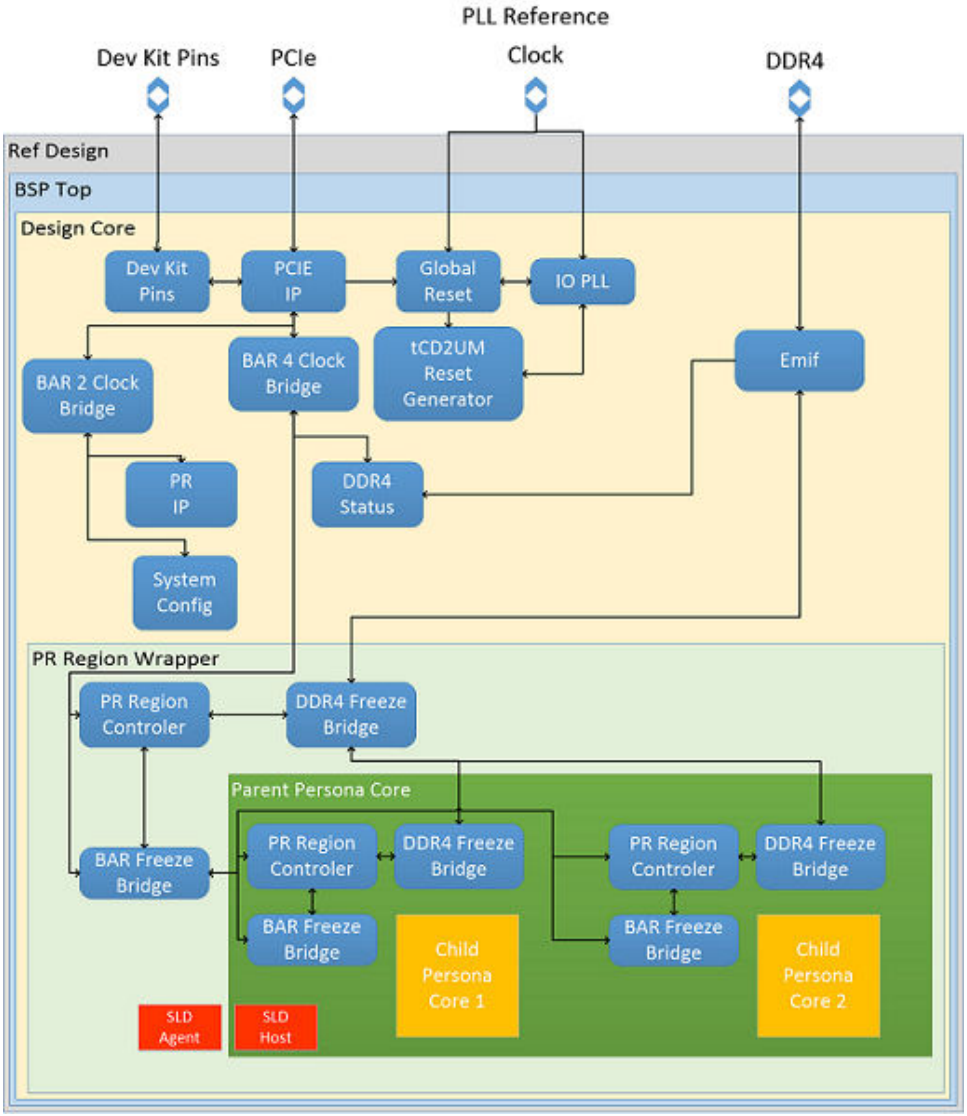
- Creating a Partial Reconfiguration Design
  パーシャル・リコンフィグレーション・デザイン・フローについての完全な情報を確認することができます。
- Hierarchical Partial Reconfiguration of a Design on Arria 10 GX FPGA Development Board
  階層的パーシャル・リコンフィグレーション・デザインの作成についてのステップごとのチュートリアルです。
- Arria 10 FPGA Development Kit User Guide
  Arria 10 GX FPGA 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—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® Memory-Mapped (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。

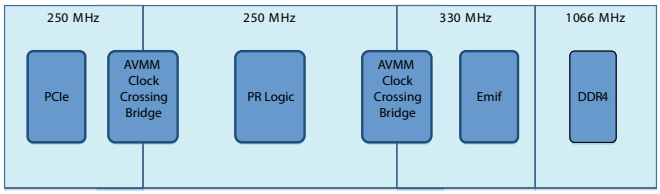
図 -1: Arria 10 PCIe リファレンス・デザインのブロック図



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 で動作する外部メモリー・インターフェイス (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、2 つの Avalon -MM マスター・インターフェイスを介して接続します。このような Avalon -MM マスター・インターフェイスのベース・アドレス・レジスター (BAR) は、BAR 2 および BAR 4 です。
BAR 2 は PR ドライバーを次のコンポーネントに接続します。

- PR IP コア
- System Description ROM

BAR 4 Avalon -MM は次のコンポーネントに接続します。

- フリーズブリッジ
- PR リージョン・コントローラー
- PR リージョンの最大 8 キロバイト (KB) のメモリー

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	ベース	エンド
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

EMIF IP は DDR4 キャリブレーションのステータスを提供します。初期化中に、EMIF IP は DDR4 インターフェイスをリセットするためのトレーニングを実行します。EMIF キャリブレーション・フラグは、トレーニングの成功または失敗をホストに報告します。DDR4 トレーニングが成功しなかった場合、ホストは必要な措置を取ります。
次の表に、EMIF IP 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

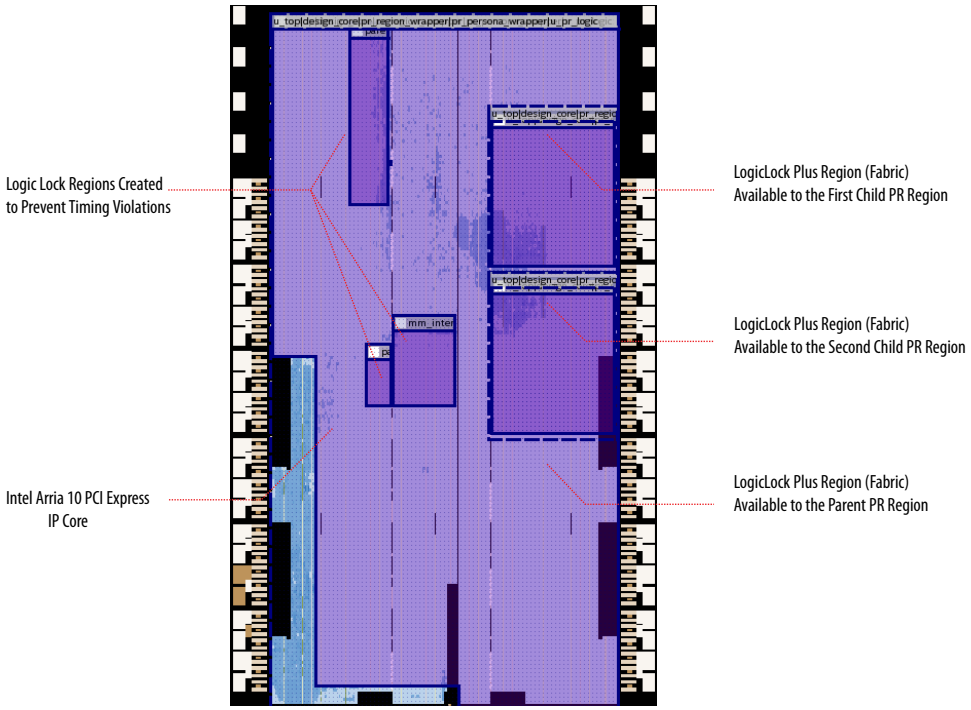
アドレスマップ	ベース	エンド
DDR	0x0000_0000	0x7fff_ffff

PR ロジックは、Avalon -MM マスター・インターフェイスを使用して、2 ギガバイト (GB) の DDR4 メモリースペースにアクセスします。

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。

図 -3: リファレンス・デザインのフロアプラン



注意:

フロアプランニングについての詳細は、*インテル Quartus Prime 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

このリファレンス・デザインを使用するには次のハードウェアおよびソフトウェア・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 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
- 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 プロ・エディション開発ソフトウェア v.18.1
- インテル FPGA ダウンロード・ケーブルドライバー
- PCIe リファレンス・デザインでこの PR に向けたオープン・ソース・ドライバーのテストに使用された CentOS 7 の検証テスト

注意:

- このリファレンス・デザインに付随する Linux ドライバーは、プロダクション・ドライバーではありません。使用するデザインを基にこのドライバーを適合させる必要があります。

1.2.2. Arria 10 GX FPGA 開発キットのインストール

Linux システムでの 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、*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

- Arria 10 FPGA Development Kit User Guide
- リファレンス・デザインの立ち上げ (9 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 ドライバーは、`debugfs` を必要とします。次のコマンドを実行して、`debugfs` が使用可能であることを確認します。

```
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:

このドライバーは、CentOS 7 のみサポートします。

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で利用可能であ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 DEVICE="a10」
   ```

   注意: Verbose messaging をイネーブルするには、`make` コマンドでオプションの VERBOSE=`true` を使用します。

   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され、かつコンピューター再起動されている場合にのみ実行されます。

Linux ドライバーのアンインストール

Linux ドライバーをアンインストールする場合は、次の手順に従ってください。

- 次のコマンドを実行します。

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

このコマンドは、ドライバーの実行を停止し、非アクティブ化します。ただし、この時点ではマシーンをリブートするとドライバーが継続してリロードされます。

- このドライバーを永久に消去するには、次のコマンドを実行してください。

```
cd /lib/modules/$(uname -r)/extra
rm -rf fpga-pcie-mod.ko
```

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_hpr` フォルダーを Linux システムのホーム・ディレクトリーに直接コピーします。

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 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_hpr/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. 接続したデバイスの JTAG ケーブル番号を引数としてスクリプトへ渡します (例、`perl flash.pl 1`)。

   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、次の 3 つの主要コンポーネントで構成されています。

- トップレベル・デザイン
- PCIe IP
- DDR4 EMIF IP

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 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、250 MHz 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 のパラメーター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
	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. 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、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 Avalon -MM スレーブは 300 MHz で動作します。

次の表に、DDR4 HILO を備えた Arria 10GX FPGA 開発キットのプリセットとは異なる Arria 10 外部メモリー・インターフェイス IP のパラメーターをリストします。

表 4. 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
- System Description 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 コアは、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。PLL がロックされた信号は、PCIe で ORed 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 PR 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ます。PR リージョン・コントローラーは、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. 親 PR リージョン

親 PR リージョンには、親 PR リージョン内に 2 つの子ペルソナが存在するため、2 組の PR リージョン・コントローラーとフリーズブリッジが含まれています。それぞれの PR リージョン・コントローラーとフリーズブリッジのセットは、1 つの子ペルソナに専用となります。

1.3.1.3.4. 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

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 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

1.3.1.3.5. 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。
Parent persona	2 つの子パーティションをインスタンス化するラッパーです。親ペルソナも、2 つの子ペルソナを独自の PR 領域コントローラー、BAR フリーズブリッジ、および DDR4 フリーズブリッジを使用してスタティック・リージョンに接続します。

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

1. リファレンス・デザインのベース・リビジョンをコンパイルするには、プロジェクト・ディレクトリー・レベルから次のコマンドを実行してください。

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c a10_pcie_devkit_cvp
```

ベースリビジョンを除くすべての実装リビジョンには、それぞれの .qsf ファイルに次の QDB ファイル・パーティション割り当てが含まれています。

```
set_instance_assignment -name QDB_FILE_PARTITION \
      output_files/a10_pcie_devkit_cvp_static.qdb -to |
```

この割り当ては、リファレンス・デザインのスタティック・リージョン・ロジックを表す .qdb ファイルを後続の PR ペルソナ実装コンパイルにインポートします。各実装リビジョンには、1 つまたは 2 つの ENTITY_REBINDING 割り当ても含まれています。この割り当ては、スタティック・リージョンの階層と PR ペルソナの階層をリンクします。たとえば、a10_pcie_devkit_cvp_ddr4_access.qsf には、次のエンティティー・リバインド割り当てが含まれています。

```
set_instance_assignment -name ENTITY_REBINDING \
      parent_persona_top -to \
      u_top|design_core|pr_region_wrapper|pr_persona_wrapper|u_pr_logi
```

詳細については、*Partial Reconfiguration User Guide* の項 *Partial Reconfiguration Design Flow* を参照してください。

2. すべての HPR でないペルソナをコンパイルするには、次のコマンドを実行してください。

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_normal_ddr4_access
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_normal_basic_arithmetic
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_normal_basic_dsp
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_normal_gol
```

3. 子 Logic Lock リージョンが、ベース・リビジョンと同じ設定でこのリビジョンで定義されていることを確認してください。
4. HPR 親ペルソナをコンパイルするには、次のコマンドを実行してください。

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_ddr4_access
```

注意: この手順を実行した後で、親 PR パーティションの .qdb ファイルを手動でエクスポートする必要があります。

すべての HPR 子実装リビジョンには、追加の QDB FILE PARTITION 割り当てが含まれています。

```
set_instance_assignment -name QDB_FILE_PARTITION \
    output_files/a10_pcie_devkit_cvp_ddr4_access_pr_partition_final.qdb -
to \
    u_top|design_core|pr_region_wrapper|pr_persona_wrapper|u_pr_logic
```

この割り当ては、HPR 親リージョンを表現している .qdb ファイルを次の HPR 子リージョンのコンパイルにインポートします。HPR 子リビジョンは 2 つの子リージョンで構成されているため、これには 2 つの ENTITY REBINDING 割り当てが含まれます。

```
set_instance_assignment -name ENTITY_REBINDING \
    basic_arithmetic_persona_top -to \
    u_top|design_core|pr_region_wrapper| \
    pr_persona_wrapper|u_pr_logic|u0|child_region_0|u_child_pr_logic


set_instance_assignment -name ENTITY_REBINDING \
    basic_arithmetic_persona_top -to \
    u_top|design_core|pr_region_wrapper|pr_persona_wrapper| \
    u_pr_logic|u0|child_region_1|u_child_pr_logic
```

5. HPR 子ペルソナをコンパイルするには、次のコマンドを実行してください。

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_basic_arithmetic
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_basic_dsp
quartus_sh --flow compile a10_pcie_devkit_cvp -c \
      a10_pcie_devkit_cvp_gol
```

関連情報

- Partial Reconfiguration Design Flow
- AN 806: インテル Arria 10 GX FPGA 開発ボードに向けた階層的パーシャル・リコンフィグレーションのチュートリアル

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`
- `fpga-configure`
- `fpga-region-controller`

このデザインには、デバイスと通信し、各ペルソナを示すための `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_fpga_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。
  - 状態を保存します。
  - AER を元の状態に戻します。
  - 状態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。
-i, --index	JTAG チェーン内のターゲットデバイスのインデックスを指定します。
-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. fpga-configure

パーシャル・リコンフィグレーションを実行するには、fpga-configure 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 コアに書き込みます。
- 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。

PCIe を介してパーシャル・リコンフィグレーションを実行するには、次のコマンドを実行してください。

```
fpga-configure -p <path-to-rbf> <region_controller_addr>
```

表 7. fpga-configure コマンドライン・オプション

オプション	説明
-p	PCIe プログラミングを介してパーシャル・リコンフィグレーションを実行します。
-d	PCIe リンクでの Advanced Error Reporting (AER) をディスエーブルします。AER は通常、PCIe リンク上に重大なエラーが存在する場合、それを直接カーネルにレポートします。PCIe リンクが完全にディセーブルされている場合、カーネルはシステムをクラッシュさせて応答します。フルチップ・コンフィグレーションは PCIe リンクを停止させてしまうので、フルチップ・コンフィグレーション実行中は AER レポートをディスエーブルする必要があります。
-e	PCIe リンクへの AER をイネーブルします。PCIe リンクのインテグリティーを確認するには、このオプションをフルチップ・コンフィグレーション後に使用します。
-r	リファレンス・デザイン内のデバッグ ROM の内容を表示します。デバッグ目的に使用します。

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。
-v, --verbose	実行中に追加情報を表示することが可能です。このオプションはデフォルトでディスエーブルされています。
-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。デフォルト値は 3 です。
-h, --help	example_host アプリケーションのヘルプを提供します。

注意: コマンドライン引数を指定せずに example_host_uio を実行すると、1、3 回のシード値が使用されます。

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・リージョンと親 PR 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 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 シェルで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 -s 1 -n 100 -v
```

このコマンドは、値が 1 の入力生成をシードし、イタレーションを 100 回実行し、現在の状況についてより多くの情報を表示します。

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

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. Programmer を使用してベース・リビジョン `.sof` ファイルをプログラミングします。PCIe が列挙可能となるようホスト PC を再起動します。PCIe デバイスとして FPGA が確実に表示されるには、Linux シェルで次の内容を入力します。

```
lspci -vvvd1172:
```

2. デザインの機能性を検証するには、たとえば次のような内容を Linux シェルで入力します。

```
./example_host_uio
```

3. 次のいずれかのシングル・ファンクション PR ペルソナでデザイン内の親 PR パーティションを置き換えるには、Linux シェルで次を入力します。

```
fpga-configure -p <rbf file from list> 10000
```

`<rbf file from list>` は次のいずれかのファイルです。

- `a10_pcie_devkit_cvp_normal_basic_arithmetic.pr_partition`
- `a10_pcie_devkit_cvp_normal_basic_dsp.pr_partition`
- `a10_pcie_devkit_cvp_normal_ddr4_access.pr_partition`
- `a10_pcie_devkit_cvp_normal_gol.pr_partition`

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

```
./example_host_uio
```

5. 2 つの子パーティションを含む親 PR パーティションをプログラムするには、Linux シェルで次を入力します。

```
fpga-configure -p a10_pcie_devkit_cvp_ddr4_access.pr_partition.rbf 10000
```

子パーシャルは両方とも DDR4 アクセスペルソナです。

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

```
./example_host_uio
```

7. さらに、それぞれの子 PR パーティションは、ペルソナの任意の組み合わせで再プログラミングが可能です。以下は、`output_files` ディレクトリーに生成されるファイルです。
   - a10_pcie_devkit_cvp_ddr4_access.pr_partition.pr_child_partition_1.rbf
   - a10_pcie_devkit_cvp_basic_dsp.pr_partition.pr_child_partition_0.rbf
   - a10_pcie_devkit_cvp_basic_dsp.pr_partition.pr_child_partition_1.rbf
   - a10_pcie_devkit_cvp_basic_arithmetic.pr_partition.pr_child_partition_0.rbf
   - a10_pcie_devkit_cvp_basic_arithmetic.pr_partition.pr_child_partition_1.rbf
   - a10_pcie_devkit_cvp_gol.pr_partition.pr_child_partition_0.rbf
   - a10_pcie_devkit_cvp_gol.pr_partition.pr_child_partition_1.rbf

   各ビットストリーム・ファイルは特定の子 PR リージョンに固有であり、互換性はありません。たとえば、`*.pr_child_partition_0.rbf` ファイルは、子 PR リージョン 0 にのみ互換性があり、1 とは互換性がありません。

8. それぞれの子リージョンをコンパイルするには、Linux シェルで次の内容を入力します。

   子リージョン 0 の場合

```
fpga-configure -p <persona>.pr_partition.pr_child_partition_0.rbf 10
```

   子リージョン 1 の場合

```
fpga-configure -p <persona>.pr_partition.pr_child_partition_1.rbf 20
```

   より大きい PR リージョンに向けた PR リージョン・コントローラーは、アドレス 0x10000 です。より小さい子 PR リージョンに向けた PR リージョン・コントローラーは、それぞれアドレス 0x10 とアドレス 0x20 です。

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_hpr` フォルダーに移動します。

```
cd a10_pcie_devkit_cvp_hpr
```

2. `pr_logic_impl_template.qsf.template` 実装リビジョン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
```

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. 次の割り当てを含めるよう persona_impl_revision_name.qsf を変更します。

```
set_global_assignment -name TOP_LEVEL_ENTITY a10_pcie_ref_design
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>
set_instance_assignment -name QDB_FILE_PARTITION \
      a10_pcie_devkit_cvp_static.qdb -to | \

set_instance_assignment -name ENTITY_REBINDING \
      <custom_persona_top_level_entity> -to u_top|design_core|
pr_region_wrapper|pr_persona_wrapper|u_pr_logic
```

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. 実装リビジョンを含めるよう `a10_pcie_devkit_cvp.qpf` プロジェクトファイルを更新します。

```
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 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. AN 813: PCI Express リファレンス・デザインを使用した Arria 10 デバイスの階層的なパーシャル・リコンフィグレーション 改訂履歴

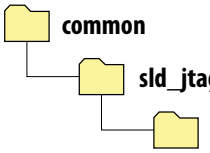
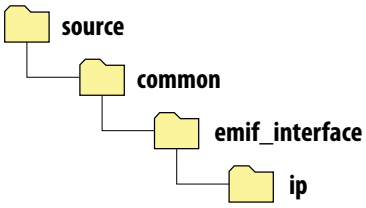
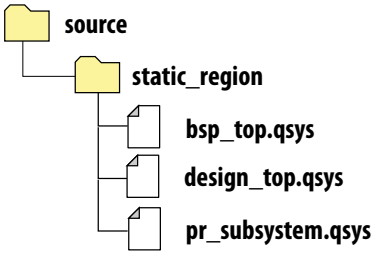
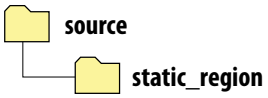
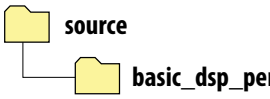
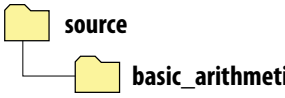
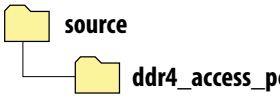
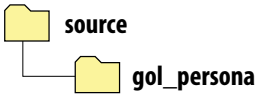
ドキュメント・バージョン	インテル Quartus Prime のバージョン	変更内容
2018.09.24	18.1.0	• スタティック・リージョンの最終的なスナップショットの手動エクスポートが必要であるという内容を削除するために *リファレンス・デザインのコンパイル* を更新しました。 • 編集上の軽微な変更を行いました。
2018.07.03	18.0.0	*リファレンス・デザインのコンパイル* を `.qdb` ファイルのエクスポートに使用する正しいコマンドで更新しました。
2018.05.07	18.0.0	• コンパイルフローを変更しました。 • 編集上の軽微な変更を行いました。
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。
2017.05.22	17.0.0	初版

A. リファレンス・デザイン・ファイル

表 9. リファレンス・デザイン・ファイルのリスト

タイプ	ファイル/フォルダー	説明
IP ファイル	source static_region ip bsp_top	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。
	source static_region ip design_top	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、System Description 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。
	source static_region ip pr_subsystem	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。
	common reg_file ip reg_file	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。
	common sld_jtag_host ip	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...

タイプ	ファイル/フォルダー	説明
		PR リージョンのシグナルタッピング用の JTAG SLD エージェントが含まれます。これらのファイルはスタティック・リージョンに適用可能です。
		PR ペルソナ内の EMIF ロジック用の IP ファイルが含まれています。
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。
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 DSP 用のすべてのソースファイルが含まれています。
		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。

continued...

タイプ	ファイル/フォルダー	説明
	source parent_persona	親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。
	source templates	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	avalon_config.hex	System Description ROM に使用されます。
Synopsys デザイン制約ファイルです。	a10_pcie_devkit_cvp.sdc	デザインの合成制約です。
	auxiliary.sdc	例外を提供します。
	jtag.sdc	`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	a10_pcie_devkit_cvp.qpf	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	a10_pcie_devkit_cvp.qsf	シングル DDR4 access ペルソナ用のベースリビジョン設定ファイルです。
	a10_pcie_devkit_cvp_ddr4_access.qsf	2 つの DDR4 access ペルソナを持つ親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	2 つの Basic DSP ペルソナを持つ親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	2 つの Basic arithmetic ペルソナを持つ親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	2 つのライフゲーム・ペルソナを持つ親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_normal_ddr4_access.qsf	親ペルソナを持たないシングル DDR4 アクセス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_normal_basic_arithmetic.qsf	親ペルソナを持たないシングル 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_normal_gol.qsf	親ペルソナを持たないシングル・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。