XAPP1220 (v1.2.1) 2025 年 5 月 28 日

UltraScale FPGA の BPI コンフィギュレーションおよびフラッシュ プログラム

著者: Stephanie Tapp、Ryan Rumsey

この資料は表記のバージョンの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。資料によっては英語版の更新に対応していないものがあります。日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

概要

同期読み出しおよび EMCCLK (外部マスター コンフィギュレーション クロック) を使用する AMD UltraScale™ アーキテクチャのマスター BPI (バイト ペリフェラル インターフェイス) コンフィギュレーション モードは、大容量の不揮発性パラレル NOR フラッシュ ストレージを利用し、マスター SPI (シリアル ペリフェラル インターフェイス) を使用するコンフィギュレーションよりもコンフィギュレーション時間を短縮できます。

このアプリケーション ノートでは、UltraScale FPGA とパラレル NOR フラッシュ (BPI フラッシュ メモリ) のインターフェイス接続、Vivado™ Design Suite 2014.4 を使用したフラッシュのプログラム手順、BPI コンフィギュレーション モードのプロセスについて説明します。このアプリケーション ノートを使用する前に、FPGA コンフィギュレーションについて説明している『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1]、およびデバイスのプログラム フローについて説明している『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 2] の内容を十分理解している必要があります。

重要: 同期読み出しを使用するマスター BPI コンフィギュレーション モードは、フラッシュ デバイスの入手可能性を踏まえ、新規デザインには推奨していません。サポート対象のフラッシュ デバイスの提供状況については、フラッシュ デバイスのサプライヤーに確認することを推奨します。サポートしているフラッシュ デバイスについては、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

注記: このアプリケーション ノートは、AMD Spartan™ UltraScale+™ FPGA には適用されません。

はじめに

UltraScale FPGA は、電源投入後にコンフィギュレーション ビットストリームをロードする必要があります。パラレル NOR フラッシュは、帯の広い x16 データ バスによって SPI フラッシュ メモリよりも高速なコンフィギュレーションが可能なため、ビットストリームの格納や送信に最適です。ランダム アクセスの不揮発性アプリケーション データの格納にパラレル NOR フラッシュを使用するシステムでは、コンフィギュレーション データを 1 つのメモリ デバイスに集約できるという利点もあります。

このアプリケーション ノートでは、パラレル NOR フラッシュを使用した 2 つの一般的なプログラム フローについて説明します。図 1 に、これらのフローを示します。

- Vivado Design Suite を使用し、JTAG ケーブル経由で FPGA を介するフラッシュの間接プログラム
- パラレル NOR フラッシュに格納されたビットストリームからの FPGA の BPI コンフィギュレーション

AMD アダプティブ コンピューティングは、従業員、顧客、パートナーそれぞれが認められ、受け入れられていると実感できる環境作りに取り組んでいます。その一環として、製品資料およびその他関連資料に含まれる非包括的な用語/表現を排除していきます。ソフトウェアおよび IP に組み込まれている用語を含め、人を差別、疎外したり、歴史的な偏見を増長する可能性のある表現をなくすための社内的取り組みが始まっています。該当表現を改め、進化する業界標準に則った取り組みを進めていますが、旧製品資料には配慮に欠ける表現が残っている可能性があります。詳細は、このページを参照してください。

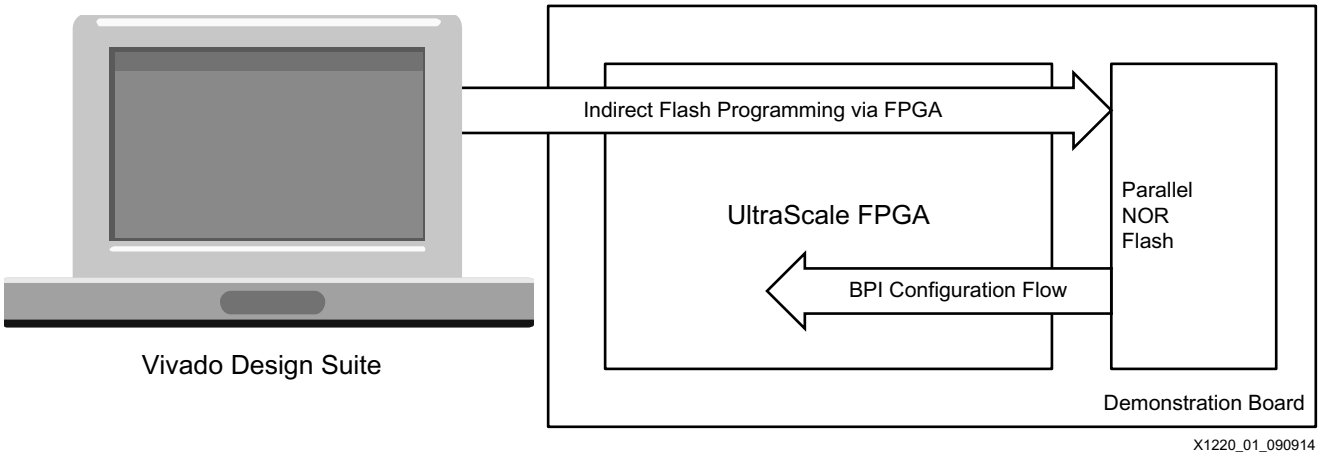

図 1: BPI コンフィギュレーションおよびフラッシュ プログラム フロー

2 つのフローおよび FPGA の BPI コンフィギュレーションを正常に完了するには、次の主な手順を実行する必要があります。

1. FPGA とパラレル NOR フラッシュを正しい方法で接続します。
2. Vivado Design Suite を使用してビットストリーム (`.bit`) ファイルを作成します。
3. Vivado Design Suite を使用してフラッシュのプログラム ファイル (`.mcs`) を作成します。
4. Vivado Design Suite を使用してインシステムでパラレル NOR フラッシュをプログラムします。
5. パラレル NOR フラッシュからターゲット FPGA をコンフィギュレーションします (電源を切って入れ直すか、PROGRAM_B をパルスする)。

この資料は次の各セクションで構成されます。

- 「UltraScale FPGA の BPI コンフィギュレーションおよびフラッシュ プログラム」では、このアプリケーション ノートの 2 つの基本フローについて説明します。
- 「ハードウェアと接続」では、正しいハードウェアのセットアップと FPGA の BPI コンフィギュレーション インターフェイスの接続について詳しく説明します。
- 「BPI コンフィギュレーションのファイル生成」では、FPGA のビットストリーム ファイルとパラレル NOR フラッシュのプログラム ファイルの生成に必要な手順とオプションについて説明します。
- 18 ページの「パラレル NOR フラッシュの間接プログラム」では、パラレル NOR フラッシュのプログラム手順について説明します。
- 「BPI コンフィギュレーションのシーケンス」では、同期読み出しを使用する BPI コンフィギュレーション プロセスとコンフィギュレーション時間の概算について説明します。
- 「コンフィギュレーション時間」には、所定の FPGA およびコンフィギュレーション周波数でのコンフィギュレーション時間の概算に必要な詳細情報が記載されています。
- 「デバッグ ガイダンス」では、BPI コンフィギュレーション モードをインプリメントする際によくある見落としを回避するための主要なヒントをまとめています。

UltraScale FPGA の BPI コンフィギュレーションおよびフラッシュ プログラム

コンフィギュレーションとは、フラッシュ デバイスまたはマイクロプロセッサなどの外部ソースを使用して、FPGA にコンフィギュレーション データをダウンロードするプロセスです。BPI コンフィギュレーション モードでは、パラレル NOR フラッシュのアドレス信号、x16 または x8 データ バス信号、制御信号に直接 FPGA を接続して、格納されているビットストリームを読み出すことができます。UltraScale FPGA の BPI コンフィギュレーション インターフェイスは、2 つのフラッシュ読み出しオプション (非同期または同期) と 2 つのコンフィギュレーション クロック オプション (内部コンフィギュレーション クロック (CCLK) または外部マスター コンフィギュレーション クロック (EMCCLK)) をサポートします。このアプリケーション ノートでは、同期読み出しオプションと EMCCLK の例を示します。このモードでは、非同期読み出しと CCLK を組み合わせるよりもはるかに高速にコンフィギュレーション データを送信できます。コンフィギュレーション時間の概算については、25 ページの「コンフィギュレーション時間」を参照してください。

Vivado Design Suite は、パラレル NOR フラッシュと FPGA 間の既存のコンフィギュレーション接続を使用して、パラレル NOR フラッシュをインシステムで間接的にプログラムする機能を備えています。この機能は、間接プログラム ビットストリームを使用し、JTAG を介して FPGA をコンフィギュレーションします。ビットストリームは、JTAG プログラミング ケーブルとパラレル NOR フラッシュ インターフェイス間の FPGA を介したパスを実現します。Vivado のインシステム プログラム機能は、プロトタイプ段階における反復作業を要するデザインのテストやデバッグを可能にしますが、量産品のプログラムを目的とするものではありません。量産品のプログラムには、BPM Microsystems 社や Data I/O 社などのベンダーが提供するプログラマ向けソリューションを使用してください。

パラレル NOR フラッシュの選択

パラレル NOR フラッシュ デバイスを選択する際は、アプリケーションで必要な格納容量、ボードのスペース要件に合ったパッケージ タイプ、コンフィギュレーション時間に応じたデータ バス幅、フラッシュの I/O 電圧範囲など、コンフィギュレーション ソースに関するいくつかの要素を考慮する必要があります。詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] を参照してください。サポートしているフラッシュ デバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 2] を参照してください。

このアプリケーション ノートでは、最も高速なコンフィギュレーションが可能で、必要な集積度を備えた、Micron 28F00AG18F (MT28GU01GAAA1E) フラッシュを使用します。

ハードウェアと接続

次のハードウェアを使用して、UltraScale FPGA の BPI コンフィギュレーション (同期読み出しと EMCCLK を使用) およびフラッシュのプログラムについて説明します。

- AMD Virtex™ UltraScale XCVU095
- Micron パラレル NOR フラッシュ 28F00AG18F (MT28GU01GAAA1E)
- Digilent USB ケーブルまたは AMD プラットフォーム ケーブル USB (サポートしているケーブルのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照 [参照 2])

注記: Micron パラレル NOR フラッシュ 28F00AG18F の製品名は、MT28GU01GAAA1E に変更されました。フラッシュ デバイスの機能に変更はありませんが、アドレス信号名が変更されています。ここでは、28F00AG18F の信号名を使用しています。

パラレル NOR フラッシュ接続の基本

図 2 に、FPGA と Micron パラレル フラッシュ デバイス間に EMCCLK インターフェイスを使用する BPI コンフィギュレーションに関連する信号を示します。表 1 で、各信号について説明します。このアプリケーション ノートで説明する Micron パラレル NOR フラッシュは、16 ビット データ バス、26 ビット アドレス バス、および制御信号を使用します。

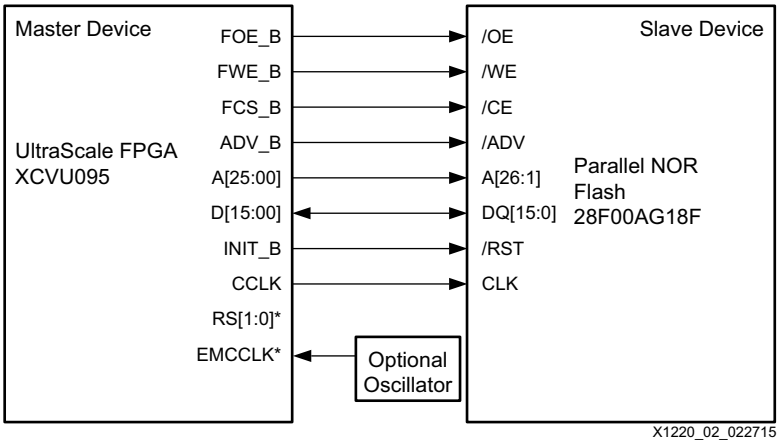

図 2: パラレル NOR フラッシュと FPGA の基本的な接続

注記: RS[1:0] および EMCCLK 信号は、オプションの高度な機能です。

BPI コンフィギュレーション インターフェイス

図 3 に、BPI コンフィギュレーションに関連する UltraScale FPGA のピンを示します。

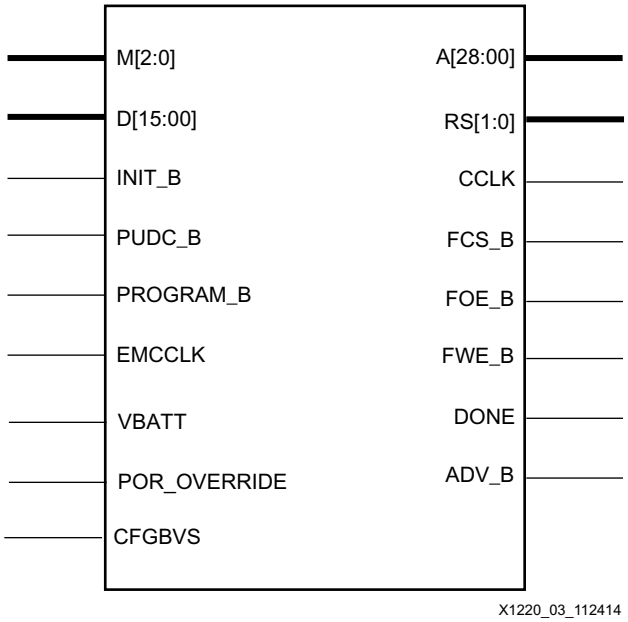

図 3: マスター BPI コンフィギュレーション モード インターフェイス

表 1 に、BPI コンフィギュレーション インターフェイス信号の機能の説明を示します。ピンの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』の表 1-6 「コンフィギュレーション ピンの説明」(UG570) [参照 1] および Micron 社の 28F00AG18F フラッシュ データシート [参照 3] を参照してください。

表 1: UltraScale FPGA の BPI コンフィギュレーション信号の説明

UltraScale FPGA ピン名	方向	説明
A[28:00]	出力	アドレス バス – パラレル NOR (BPI) フラッシュ メモリへの出力アドレスです。A00 がアドレスの最下位ビットです。FPGA の A[28:00] ピンをパラレル NOR フラッシュのアドレス ピンに接続します。FPGA の A00 ピンは、使用するデータ バス幅に応じてフラッシュの最下位アドレス入力ピンに正しく接続してください。パラレル NOR フラッシュおよび使用するデータ バス幅に応じて、フラッシュの最下位アドレス ビットは A1、A0、A-1 のいずれかとなります。パラレル NOR フラッシュのアドレス バス幅を超える上位アドレスの多目的ピンは、コンフィギュレーション中および Vivado Design Suite によるフラッシュの間接プログラム中も駆動されていることに注意してください。コンフィギュレーション後にこれらのピンを I/O として使用する場合は、このことを考慮に入れる必要があります。
RS[1:0]	出力	リビジョン セレクト – 複数ビットストリームのアプリケーションでリビジョンの選択に使用され、フォールバック機能を提供します。コンフィギュレーション エラーを検出した場合は Low に駆動され、フォールバック ビットストリームが読み込まれます。リビジョン管理の詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] を参照してください。

表 1: UltraScale FPGA の BPI コンフィギュレーション信号の説明 (続き)

UltraScale FPGA ピン名	方向	説明
CCLK	出力	コンフィギュレーション クロック – JTAG を除くすべてのコンフィギュレーション モードでは、この信号をデフォルトのコンフィギュレーション クロック ソースとして使用します。マスター BPI コンフィギュレーション モードでは、CCLK は出力となります。BPI の非同期読み出しモード中は、CCLK はパラレル NOR フラッシュに直接クロックを供給するのではなく、アドレスおよびサンプル読み出しデータを生成するために FPGA 内部で使用されます。BPI の同期読み出しモード中は、CCLK をパラレル NOR フラッシュへ直接接続し、データの逐次出力用にクロックを供給します。
FCS_B	出力	フラッシュ チップ セレクト (BAR) – アクティブ Low のフラッシュ チップ セレクト出力です。コンフィギュレーション中はアクティブにトグルされます。
FOE_B	出力	フラッシュ出力イネーブル (BAR) – アクティブ Low のフラッシュ出力イネーブルです。コンフィギュレーション中はアクティブにトグルされます。
FWE_B	出力	フラッシュ書き込みイネーブル (BAR) – コンフィギュレーション中はアクティブにトグルされます。
DONE	双方向	完了 – DONE ピンが High になると、コンフィギュレーション シーケンスが完了したことを表します。デフォルトでは、DONE 出力はオープン ドレインです。 **注記**: DONE にはデフォルトで約 10 kΩ の内部プルアップ抵抗があります。必須ではありませんが、4.7 kΩ の抵抗回路を外付けすることを推奨します。
ADV_B	出力	アドレス有効 (BAR) – アクティブ Low のアドレス有効出力です。マスター BPI コンフィギュレーション モードの同期読み出しオプションで使用します。同期読み出しモード時に、アドレスが有効であることを伝えるために必要な信号です。非同期読み出しモードの場合は、Low 駆動する必要があります。
M[2:0]	入力	コンフィギュレーション モード – コンフィギュレーション モードを決定するモード ピンです。M[2:0] = 010 の場合、マスター BPI コンフィギュレーション モードになります。
D[15:00]	双方向	データ バス – この 16 ビット データ バスは、FPGA CCLK の立ち上がりエッジでサンプルされます。このバスでデータがフラッシュから読み出され、コンフィギュレーション コントローラーから同期読み出しコマンドが発行された場合には、このバスにフラッシュの読み出しコンフィギュレーション レジスタへの書き込みコマンドが送信されます。FPGA は、D[07:00] (x8 バス幅) のみが使用されるか、より幅広い (x16) データ バス幅が使用されるかを判断する自動検出パターンを検出するための D[07:00] をモニターします。データ バスとして使用するピンを、パラレル NOR フラッシュの対応するデータ ピンに接続してください。
INIT_B	双方向	初期化 (BAR) – アクティブ Low の FPGA 初期化ピンまたはコンフィギュレーション エラー信号です。FPGA がコンフィギュレーション リセット状態にある場合や、コンフィギュレーション メモリを初期化 (クリア) 中の場合、またはコンフィギュレーション エラーを検出した場合に、このピンを Low に駆動します。FPGA の初期化プロセスが完了すると、INIT_B はハイ インピーダンスとなります。この時点で INIT_B が High になるように外部プルアップ抵抗を接続してください。初期化プロセス完了後、INIT_B 入力で High が検出されると FPGA は M[2:0] ピンの設定に従って残りのコンフィギュレーション シーケンスを実行します。コンフィギュレーション完了後は、FPGA がコンフィギュレーション エラーを検出したかどうかを示す目的で INIT_B をオプションで使用できます。
PUDC_B	入力	コンフィギュレーション中のプルアップ (BAR) – アクティブ Low 入力で、電源投入後およびコンフィギュレーション中に SelectIO™ ピンの内部プルアップ抵抗を有効にします。Low にすると、各 SelectIO ピンの内部プルアップ抵抗が有効になります。High にすると、各 SelectIO ピンの内部プルアップ抵抗が無効になります。PUDC_B は直接または ≤ 1 kΩ の抵抗を介して V_{CCO_0} または GND に接続してください。
PROGRAM_B	入力	プログラム (BAR) – アクティブ Low の非同期フルチップ リセットです。

表 1: UltraScale FPGA の BPI コンフィギュレーション信号の説明 (続き)

UltraScale FPGA ピン名	方向	説明
EMCCLK	入力	外部マスター コンフィギュレーション クロック – EMCCLK コマンドがビットストリーム ヘッダーから読み出されると、この入力に外部クロックが供給され、FPGA コンフィギュレーション コントローラーが CCLK (内部コンフィギュレーション クロック) ではなく、このクロックを使用するように切り替わります。EMCCLK を使用する場合、外部オシレーターによって偏差が決定されるため、コンフィギュレーション時間がより正確に予測できます。
VBATT	N/A	バックアップ バッテリ電源 – AES 復号化キーを格納する、FPGA の内部揮発性メモリ用電源です。ビットストリームを暗号化した場合、FPGA の電源がオフの間、復号化キーを格納した揮発性メモリの内容を保持するためにこのピンをバックアップ バッテリに接続する必要があります。揮発性のキー格納領域にある復号化キーを使用する必要がない場合は、GND または V_{CCAUX} に接続してください。
POR_OVERRIDE	入力	パワーオン リセット オーバーライド – UltraScale ファミリ データシート『Kintex UltraScale FPGA データシート: DC 特性および AC スイッチ特性』(DS892) [参照 4] および 『Virtex UltraScale FPGA データシート: DC 特性および AC スイッチ特性』(DS893) [参照 5] で指定されている、電源投入から INIT_B の立ち上がりまでのパワーオン リセットの時間 (T_{POR}) を短縮します。T_{POR} 時間を短縮する必要がある場合は、このピンを V_{CCINT} に直接接続します (ただしコンフィギュレーション データ ソースの電源投入タイミングが短縮した遅延時間をサポートしている必要がある)。標準の T_{POR} 遅延でよければ、GND に接続します。 電源投入にから起動までの時間が重視されるアプリケーションでは、UltraScale FPGA の POR_OVERRIDE 機能の使用を検討します。アプリケーションで電源の最大立ち上がり時間が UltraScale データシートの T_{POR} 仕様を満たす場合は、この信号を V_{CCINT} に接続してビルトイン遅延を短縮できます。 **注意**: コンフィギュレーション前およびコンフィギュレーション中は、このピンをフロートにしないでください。必ず V_{CCINT} または GND に接続してください。V_{CCO_0} には接続しないでください。
CFGBVS	入力	コンフィギュレーション バンク電圧の選択 - コンフィギュレーション専用バンク 0 の I/O 電圧動作範囲と電圧許容範囲を決定します。バンク 65 が HR I/O バンクである場合は、バンク 65 のコンフィギュレーション ピンの I/O 電圧動作範囲と電圧許容範囲も決定します。CFGBVS はバンク電圧の要件に従って High または Low に接続してください。バンク 0 の V_{CCO_0} 電源が 2.5V または 3.3V の場合は、このピンを High (つまり V_{CCO_0}) に接続する必要があります。バンク 0 の V_{CCO_0} が 1.8V 以下の場合にのみ、Low (つまり GND) に接続します。バンク 65 をコンフィギュレーションで使用する場合、電圧はバンク 0 と同じにする必要があります。 **注意**: デバイスの損傷を防ぐため、このピンは必ず V_{CCO_0} または GND のいずれかに正しく接続してください。

図 4 に、x16 同期読み出しモードに必要なマスター BPI コンフィギュレーション インターフェイスの接続を示します。Vivado ツールのフラッシュ プログラム機能をサポートする、JTAG ポート信号接続も図中に含まれています。

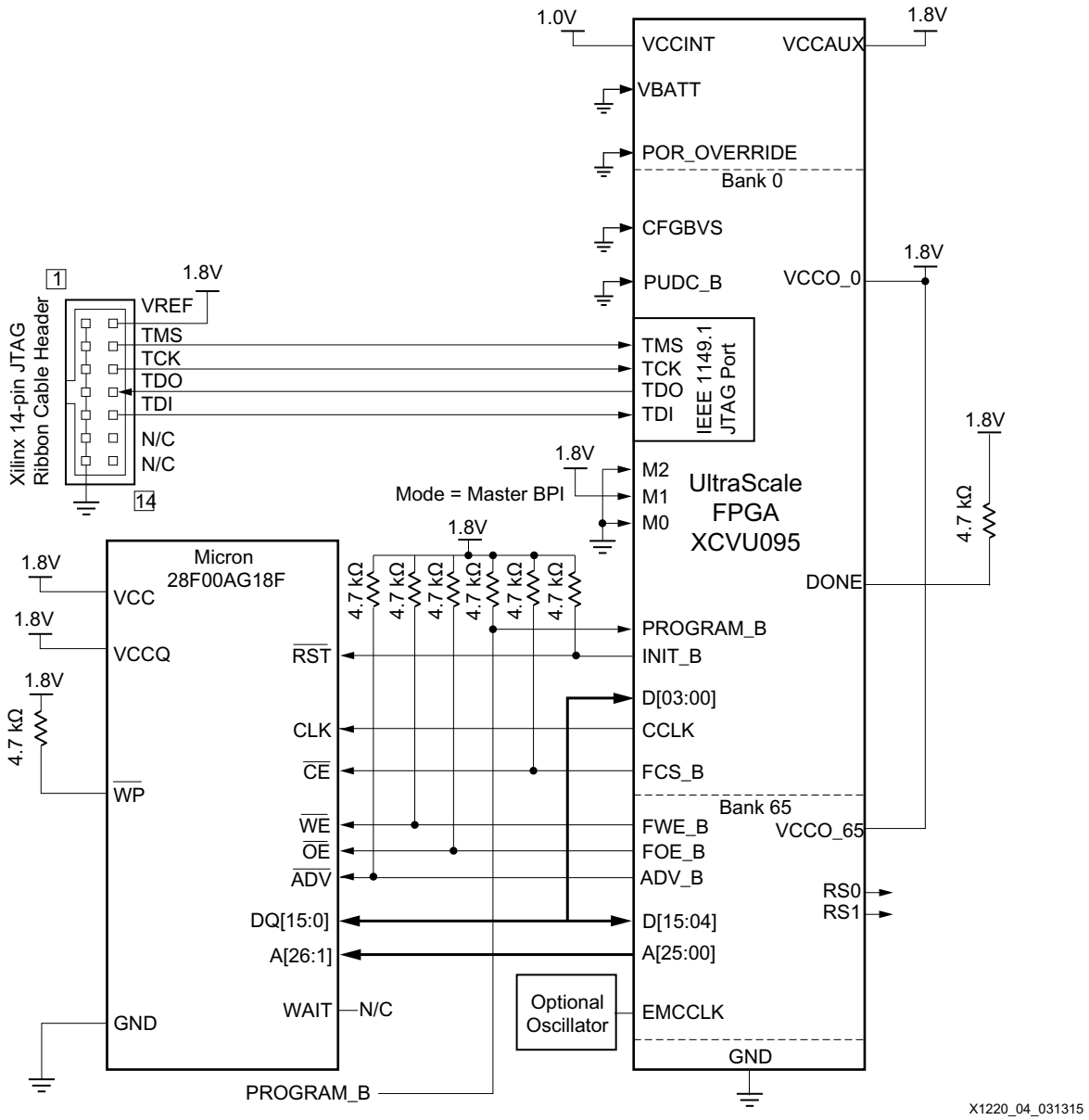

図 4: マスター BPI コンフィギュレーション x16 モードの例

図 4 について説明します。

1. この例では、バンク 0 とバンク 65 が 1.8V に設定されているため、CFGBVS はグランドに接続されます。2.5V または 3.3V を使用するデザインでは、このピンを High にする必要があります。詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] を参照してください。
2. 電源投入から起動までの時間が重視されるアプリケーションでは、UltraScale FPGA の POR_OVERRIDE 機能の使用を検討します。アプリケーションで電源の最大立ち上がり時間が UltraScale ファミリ データシートの T_{POR} 仕様 ([参照 4] および [参照 5]) を満たす場合は、この信号を V_{CCINT} に接続してビルトイン遅延を短縮できます。
3. Vivado ツールのフラッシュ プログラム機能には、UltraScale FPGA の JTAG 信号 (TCK、TMS、TDI、TCK) が必要です。JTAG インターフェイスは、数多くのアプリケーション セットアップで広く使用されるデバッグ インターフェイスでもあります。フラッシュを間接的にプログラムする手順については、18 ページの「パラレル NOR フラッシュの間接プログラム」を参照してください。

4. EMCCLK の最大周波数は、ターゲットとなるフラッシュおよび FPGA によって異なります。特定のセットアップの最大周波数を確認する場合は、25 ページの「コンフィギュレーション時間」を参照してください。
5. RS[1:0] は、リビジョン管理用のオプション ピンです。リビジョン管理が必要なアプリケーションでは、FPGA の RS[1:0] ピンをフラッシュの上位 2 本のアドレス ピンに接続します。
6. UltraScale FPGA の V_{CCO_0} 電源電圧は、パラレル フラッシュの V_{CCQ} と合わせる必要があります。
7. フラッシュの間接プログラムおよびコンフィギュレーションを実行するには、Micron 28F00AG18F の Write Protect (/WP) 信号を適切に接続する必要があります。
8. 同期読み出しを使用する BPI コンフィギュレーションには、Micron 28F00AG18F の WAIT 出力信号は不要です。ただし、同期読み出しモードでのコンフィギュレーション後にパラレル NOR のユーザー データへアクセスする場合は、WAIT 信号を FPGA の I/O に接続できます。WAIT の動作の詳細は、Micron 社のデータシート [参照 3] を参照してください。

BPI コンフィギュレーションのファイル生成

Vivado Design Suite を使用して UltraScale FPGA のビットストリームとフラッシュのプログラム ファイルを作成し、パラレル NOR フラッシュ デバイスを間接的にプログラムします。BPI コンフィギュレーション モード用に適切なファイルを生成するには、特殊なオプション設定が必要です。

図 5 に、フラッシュの間接プログラムに必要な Vivado ツール フローの概要を示します。このセクションで説明する主な手順は次のとおりです。

- BPI コンフィギュレーションに必要なデザイン制約
- デザインのインプリメント後に write_bitstream コマンドでビットストリームを生成する
- write_cfgmem コマンドでビットストリームを含むフラッシュ プログラム ファイルを作成する
- Vivado ハードウェア マネージャーがフラッシュのプログラム ファイルを使用してパラレル NOR フラッシュを間接的にプログラムする

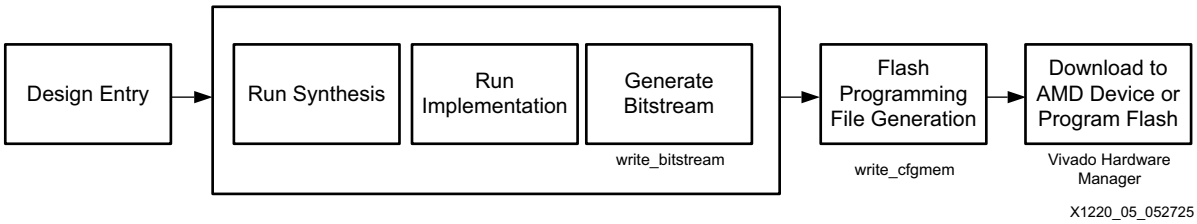

図 5: Vivado Design Suite ツールのフロー

BPI コンフィギュレーションに必要なデザイン制約

Vivado Design Suite でデザインの合成とインプリメントを実行する前に、BPI コンフィギュレーションの主要な制約を XDC ファイルに定義しておくことを推奨します。BPI コンフィギュレーション モードの重要な制約を次に示します。このアプリケーション ノートで使用するオプションについても説明します。

```
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN DIV-1 [current_design]
set_property BITSTREAM.CONFIG.BPI_SYNC_MODE TYPE1 [current_design]
set_property CONFIG_MODE BPI16 [current_design]
```

BPI コンフィギュレーションの EMCCLK オプションを使用するアプリケーションでは、EMCCLK 多目的ピンの電圧が定義されている必要があります。EMCCLK を有効にするには、多目的ピンの電圧を定義する CONFIG_VOLTAGE プロパティが必要です。

この例に示すようにバンク 0 が 1.8V で動作するには、コンフィギュレーション バンク電圧セレクト (CFGBVS) ピンが GND に接続されている必要があります。CFGBVS の詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] を参照してください。CFGBVS ピンを GND に接続するには、CFGBVS プロパティを使用します。

ビットストリームのサイズは、圧縮オプションを用いて縮小できます。圧縮率はデザインによって異なるため、保証されていません。しかし、ビットストリームのサイズを縮小すると、パラレル NOR フラッシュのプログラム時間が短縮されます。BITSTREAM.GENERAL.COMPRESS プロパティを TRUE に設定します。

EMCCLK を使用する場合は、コンフィギュレーション プロパティを設定して EMCCLK を有効にする必要があります。プロパティ BITSTREAM.CONFIG.EXTMASTERCCLK_EN を使用します。この例では EMCCLK の分割は不要なため、このプロパティは DIV–1 に設定されています。Vivado tcl コマンド list_property_values BITSTREAM.CONFIG.EXTMASTERCCLK_EN [current_design] は、利用可能な設定を返します。

最後に、コンフィギュレーション時間の短縮のために同期読み出しを使用する場合は、プロパティ BITSTREAM.CONFIG.BPI_SYNC_MODE を設定する必要があります。この例では、このプロパティは 28F00AG18F フラッシュを表す TYPE1 に設定されます。

BPI コンフィギュレーション用のビットストリーム

このセクションでは、Vivado ツールを使用して、インプリメント済みのデザインから BPI コンフィギュレーション用のビットストリームを生成する方法を例示します。次の 2 つのビットストリーム生成手法について後続のセクションで説明します。

- Vivado 統合設計環境 (IDE) フロー (プロジェクト モード) を使用する
- Vivado IDE Tcl コンソール (プロジェクト モード) または Tcl シェル (非プロジェクト モード) を使用する

それぞれの手法は、互いに独立して実行できます。これらの推奨ビットストリーム オプションの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 2] を参照してください。

Vivado IDE を使用したビットストリーム生成の例

ビットストリームは、Vivado IDE の [Generate Bitstream] フローから生成できます。BPI コンフィギュレーションをサポートするように、デフォルトのビットストリームの設定の一部を変更する必要があります。「BPI コンフィギュレーションに必要なデザイン制約」で説明したように、デザインを合成する前に、XDC ファイル内のビットストリーム プロパティを変更することを推奨します。合成前の編集が完了したら、Vivado Design Suite の Flow Navigator の下にある [Generate Bitstream] をクリックします。

その他に、ターゲットとなる BPI コンフィギュレーションに合わせて既存のデザインを変更する必要がある場合は、[Edit Device Properties] ダイアログ ボックスでプロパティを設定できます。これには、Vivado IDE で合成済みまたはインプリメント済みのデザインを開き、次の手順に従います。

1. Flow Navigator でビットストリームの設定を変更するには、図 6 に示すように、[Program and Debug] の下にある [Bitstream Settings] をクリックします。

   別の方法として、[Tools] → [Edit Device Properties] をクリックします。

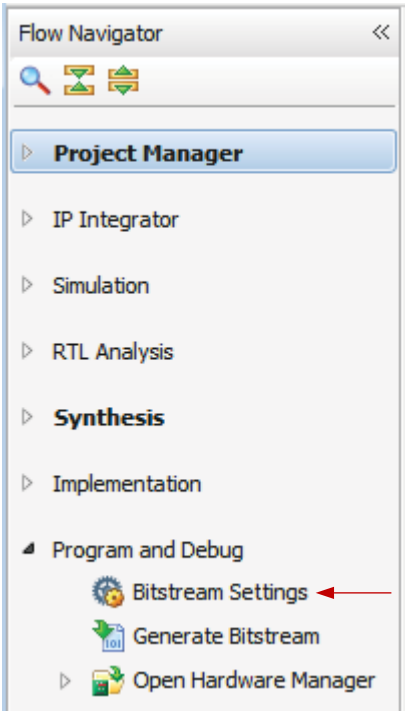

図 6: Flow Navigator ビュー

2. 図 7 に示すように、[Project Settings] で [Configure additional bitstream settings] リンクを使用してデバイス コンフィギュレーション ビットストリームの設定を編集します。このリンクは、プロジェクトで合成済みまたはインプリメント済みのデザインが開かれている場合にのみ利用可能です。

図 7: [Project Settings] の [Bitstream] ビュー

3. 図 8 の左側に示した [General] で、[Enable Bitstream Compression] を [TRUE] に設定し、全般的なプログラム時間とコンフィギュレーション時間を短縮します。この設定はオプションであり、コンフィギュレーション時間の短縮を保証するものではありません。

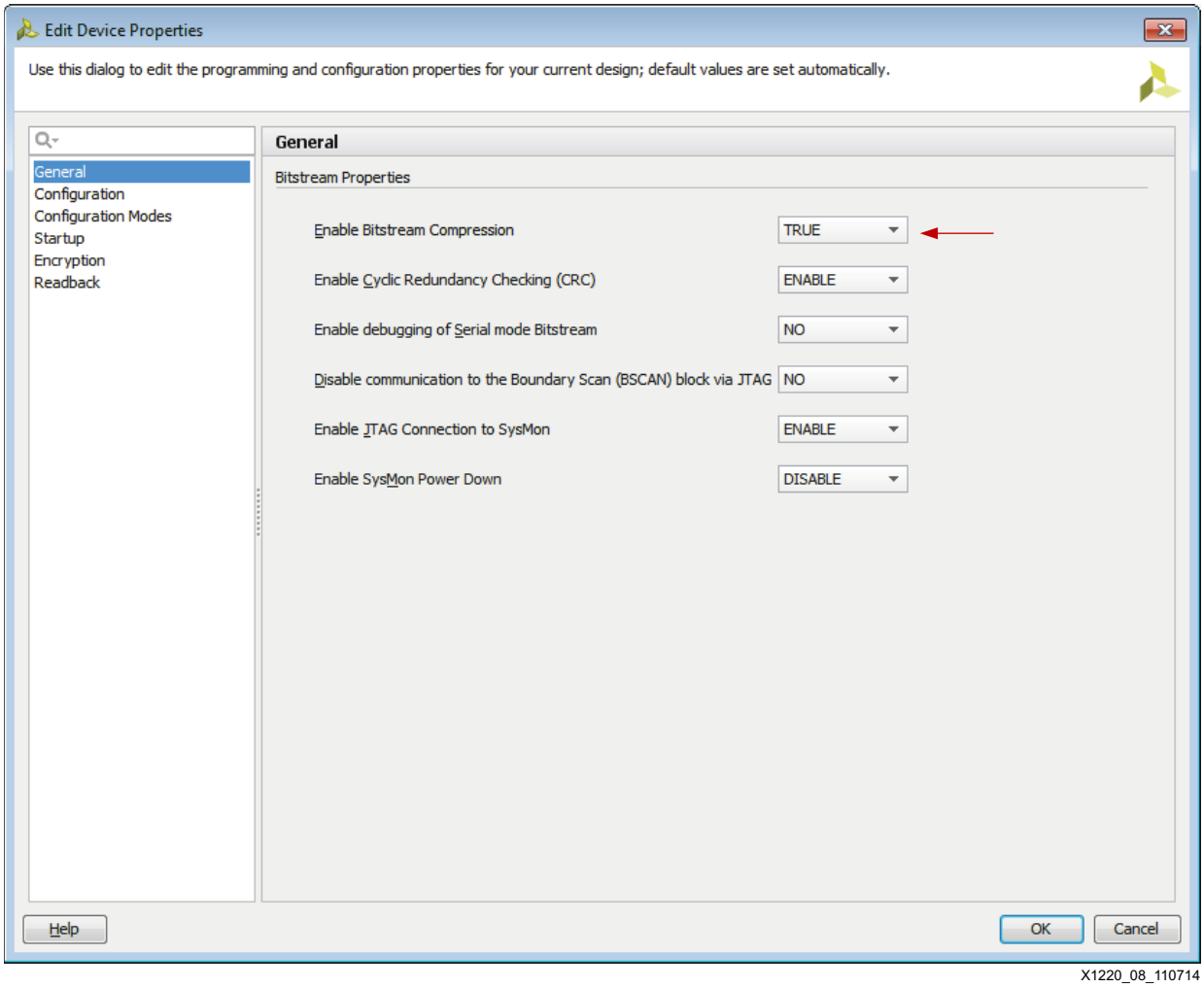

図 8: コンフィギュレーション ビットストリーム設定の [General] オプション

4. [Configuration] では、図 9 に示すようにオプションを選択して、外部コンフィギュレーション クロック (EMCCLK) を使用する同期読み出しモードの BPI コンフィギュレーション用にビットストリームを生成します。

   [Configuration Setup] で、次のように設定します。

   a. [Enable external configuration clock and set divide value] を [DIV-1] に設定します。
   b. [Configuration Voltage] を [1.8] に設定します。
   c. [Configuration Bank Voltage Selection] を [GND] に設定します。

      これらの設定により、この例で求められているようにコンフィギュレーション バンクに 1.8V が使用されます。

[BPI Configuration] で、次のように設定します。

d. [Synchronous Mode] を、パラレル NOR フラッシュ 28F00AG18F ファミリを表す [Type1] に設定します。

   プロパティの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [参照 2] を参照してください。

e. [OK] をクリックして、ビットストリームの設定変更を確定します。

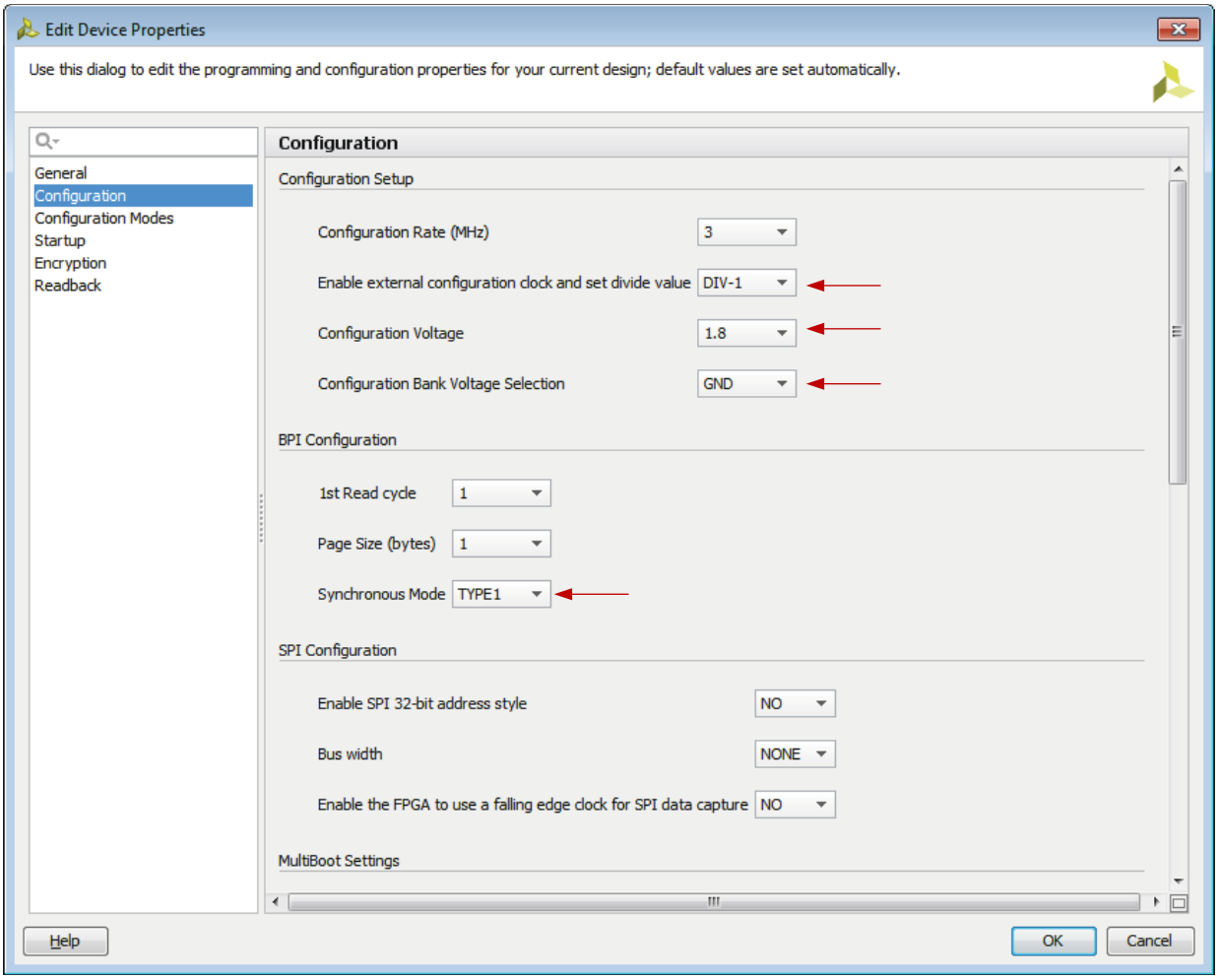

図 9: コンフィギュレーション ビットストリーム設定の [Configuration] オプション

5. 図 10 に示すように、[Configuration Modes] で [Master BPI-Up x16] がオンになっていることを確認します。

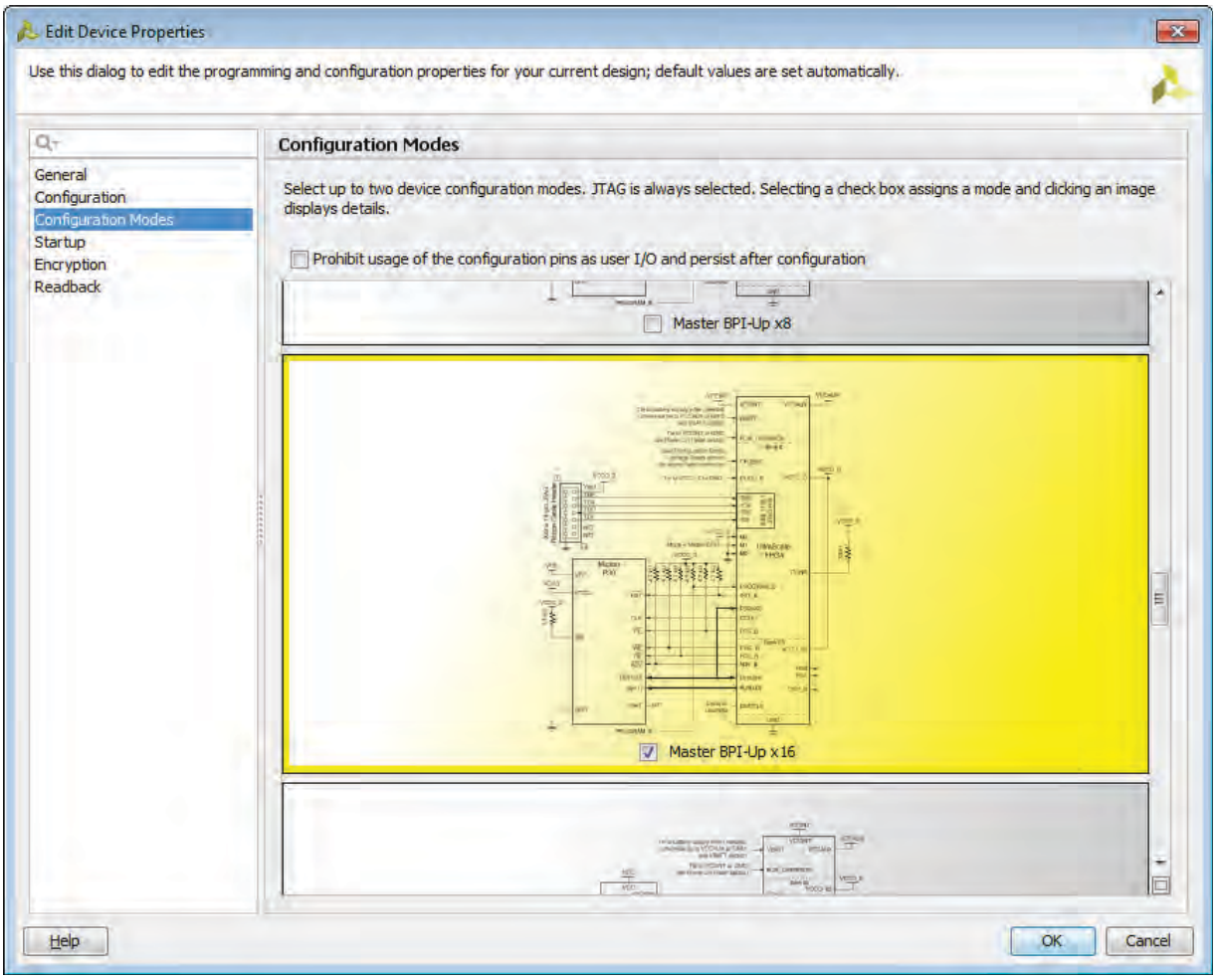

図 10: [Master BPI-Up x16] オプション

プロパティの編集が完了したら、[File] → [Save Constraints] をクリックして制約を保存します。これで、ターゲットとなる制約ファイルにプロパティが書き込まれます。インプリメンテーションを実行する前に、合成済みのデザイン内でデバイス コンフィギュレーション プロパティを変更することを推奨します。

制約ファイルの保存後に、合成済みのデザインが古くなることがあります。再合成を実行する代わりに、[Vivado tool] ビューの最下部にある [Design Runs] タブを開き、現在の合成の実行 (たとえば、synth_1) を右クリックして、[Force Up-To-Date] をクリックします。

6. 制約を保存した後、デザインのビットストリームを生成します。Flow Navigator の [Program and Debug] の下にある [Generate Bitstream] をクリックするか、[Flow] → [Generate Bitstream] をクリックします。作成されたビットストリームは、デフォルトで次の場所に置かれます。

```
<Project_Dir>\Project_Name.runs\impl_1\
```

注記: このアプリケーション ノートでは、Vivado プロジェクトのサブフォルダー ディレクトリの位置を `<Project_Dir>` で表します。

Vivado IDE Tcl コンソールを使用したビットストリーム生成の例

ビットストリームのプロパティと生成は、図 11 に示す Vivado IDE の [Tcl Console] ウィンドウか、Tcl コマンド シェルを使用して管理できます。ビットストリーム ファイルを生成する前に、デザインの制約としてビットストリーム プロパティを設定する必要があります。「BPI コンフィギュレーションのシーケンス」の説明に従って、XDC ファイルにプロパティを追加することを推奨します。既存デザインを変更する必要がある場合のコマンド ライン フローの概要を次に示します。

1. 合成済みのデザインが古くなった場合は、合成を再実行します。
2. 合成済みのデザインを開き、制約プロパティを変更します。
3. ビットストリーム プロパティを設定し、制約ファイルを保存します。
4. インプリメントを開始します。インプリメントが完了したら、インプリメント済みのデザインを開きます。
5. ビットストリームを生成します。

ビットストリームを生成するには、プロジェクトでインプリメント済みのデザインを開いておく必要があります。現在のプロジェクトのビットストリーム ファイルを書き込むコマンドは、図 11 (`write_bitstream -verbose <file name>`) を参照してください。書き込むビットストリームは、開かれているインプリメント済みデザインに基づいています。`-verbose` スイッチを指定すると、使用されるすべての `write_bitstream` オプションが表示されます。このコマンドの使用方法は、Tcl コンソールで `write_bitstream -help` と入力して確認できます。

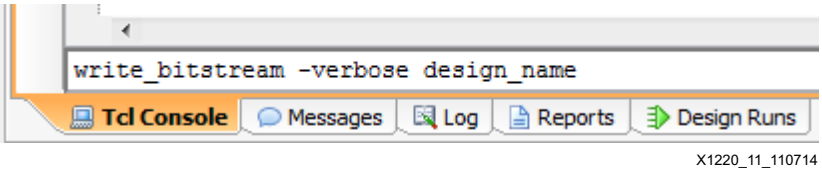

図 11: [Tcl Console] タブのコマンド例

11 ページの「Vivado IDE を使用したビットストリーム生成の例」の手順で実行される Tcl コマンドを次に示します。ターゲット デザインは合成できる状態であるとします。

```
launch_runs synth_1
open_run synth_1 -name netlist_1
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.EXTMASTERCCLK_EN DIV-1 [current_design]
set_property BITSTREAM.CONFIG.BPI_SYNC_MODE TYPE1 [current_design]
set_property CONFIG_VOLTAGE 1.8 [current_design]
set_property CFGBVS GND [current_design]
save_constraints
reset_run impl_1
launch_runs impl_1
close_design
open_run impl_1
write_bitstream Design_BIT
```

パラレル NOR フラッシュのプログラム ファイルの準備

Vivado Design Suite は、パラレル NOR フラッシュのプログラムに使用可能なフラッシュ プログラム ファイル (.mcs) を、FPGA ビットストリーム (.bit) から作成できます。フラッシュのプログラム ファイルを書き込む基本コマンドは次のとおりです。

```
write_cfgmem -format <arg> -size <arg> -interface <arg> -loadbit <arg> <file>
```

write_cfgmem コマンドの詳細は、次のように -help コマンドを使用してください。

```
write_cfgmem -help
```

Vivado IDE Tcl コンソールを使用したフラッシュ プログラム ファイル生成の例

128 MB (1 Gb) の 28F00AG18F のフラッシュ プログラム ファイルを作成するコマンドの例を次に示します。

```
write_cfgmem -format mcs -size 128 -interface BPIx16
-loadbit "up 0x0 <Project_Dir>/Project_Name.runs/impl_1/Design_Name.bit"
<Project_Dir>/Design_Name.mcs
```

この例では、ファイル フォーマットを標準の .mcs に設定し、128 MB (1 Gb) のターゲット フラッシュ サイズを使用します。-size オプションはメガバイト単位で指定します。-interface オプションは、ターゲット モードの特殊なデータ処理を指定します (この場合のデータ フォーマットは x16)。-loadbit スイッチは、ターゲットとするビットストリームの数と開始アドレス位置を指定します。

注記: コマンド ラインのファイル パスにスペースが含まれていないか注意してください。コマンド ラインでは、スペースを使用することを避けるか、または 2 つのバックスラッシュ (\\) を使用してスペースを参照からはずしてください。

Tcl コマンド ラインは、ファイル パスにスラッシュ (/) を使用する必要があります。

デフォルトでは、Vivado Design Suite はその起動元のフォルダーにファイルを書き込みます。これ以外の場所に書き込むには、次のいずれかを実行します。

- 前の例に示したように、出力ファイルのファイル パスを指定する
- コマンド cd <file path> でディレクトリを変更する:

cd および write_cfgmem を使用する例を次に示します。

```
cd <Project_Dir>
write_cfgmem -format mcs -size 128 -interface BPIx16
-loadbit "up 0x0 Project_Name.runs/impl_1/Design_Name.bit" Design_Name.mcs
```

パラレル NOR フラッシュの間接プログラム

図 12 に、Vivado ツールを使用したフラッシュの間接プログラム機能の基本セットアップを示します (UltraScale FPGA 評価ボードを使用)。Vivado ハードウェア マネージャーのユーザー インターフェイスまたは Tcl コマンドを使用して、オンボード フラッシュをプログラムできます。

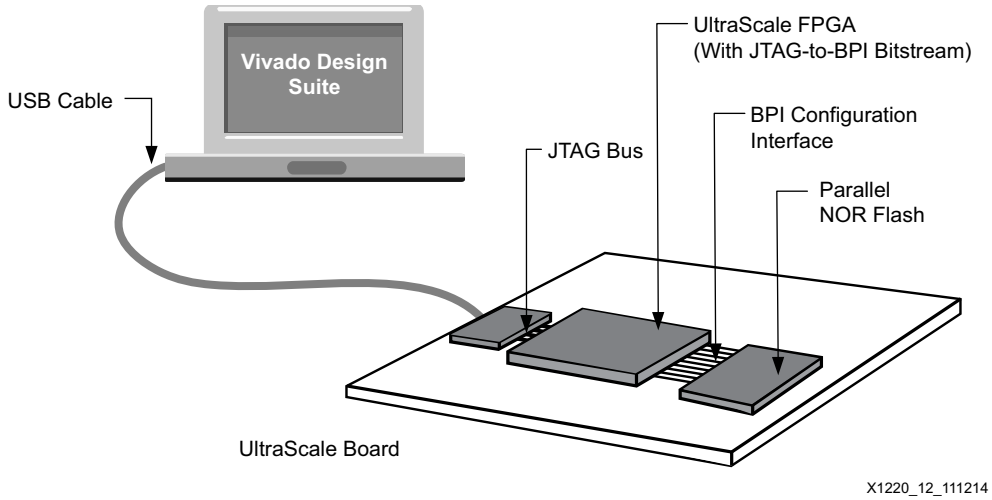

図 12: Vivado によるフラッシュの間接プログラムのセットアップ

Vivado によるフラッシュの間接プログラムのための UltraScale FPGA ボード セットアップ

Vivado Design Suite を使用してパラレル NOR フラッシュをプログラムし、UltraScale FPGA を正しくコンフィギュレーションするための基本的なボード セットアップについて簡潔に説明します。

1. モード ピン M[2:0] の設定を確認し、UltraScale FPGA の BPI コンフィギュレーション インターフェイス ピンが正しく接続されていることを確認します。図 4 を参照してください。
2. ボードの電源を入れます。
3. サポートされる JTAG Digilent USB ケーブルまたはプラットフォーム ケーブル USB II を接続します。

Vivado IDE によるパラレル NOR フラッシュのプログラム例

1. Vivado ハードウェア マネージャーを起動します。Flow Navigator の [Program and Debug] の下にある [Open Hardware Manager] をクリックします (図 13 参照)。または、[Flow] → [Open Hardware Manager] をクリックします。

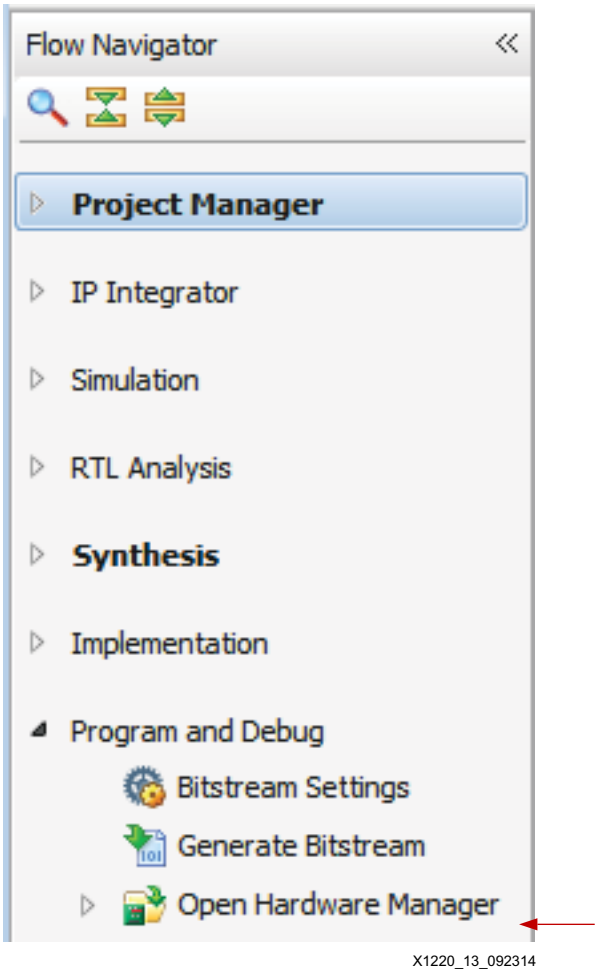

図 13: Flow Navigator ビュー

2. Flow Navigator の [Hardware Manager] の下にある [Open Target] → [Open New Target] をクリックするか、または [Tools] → [Open New Target] をクリックして、ハードウェア ターゲットを開きます。

3. Open Hardware Target Wizard が表示されます。ウィザードの指示に従って XCVU095 FPGA を選択します。図 14 に示すように JTAG 接続の周波数を選択します。この例では、デフォルト周波数の 15 MHz を使用します。[Next] を順にクリックして、ウィザードを終了します。

   ハードウェア デバイスが表示されない場合は、ケーブル接続と電源を確認します。その他のヒントについては、26 ページの「デバッグ ガイダンス」を参照してください。

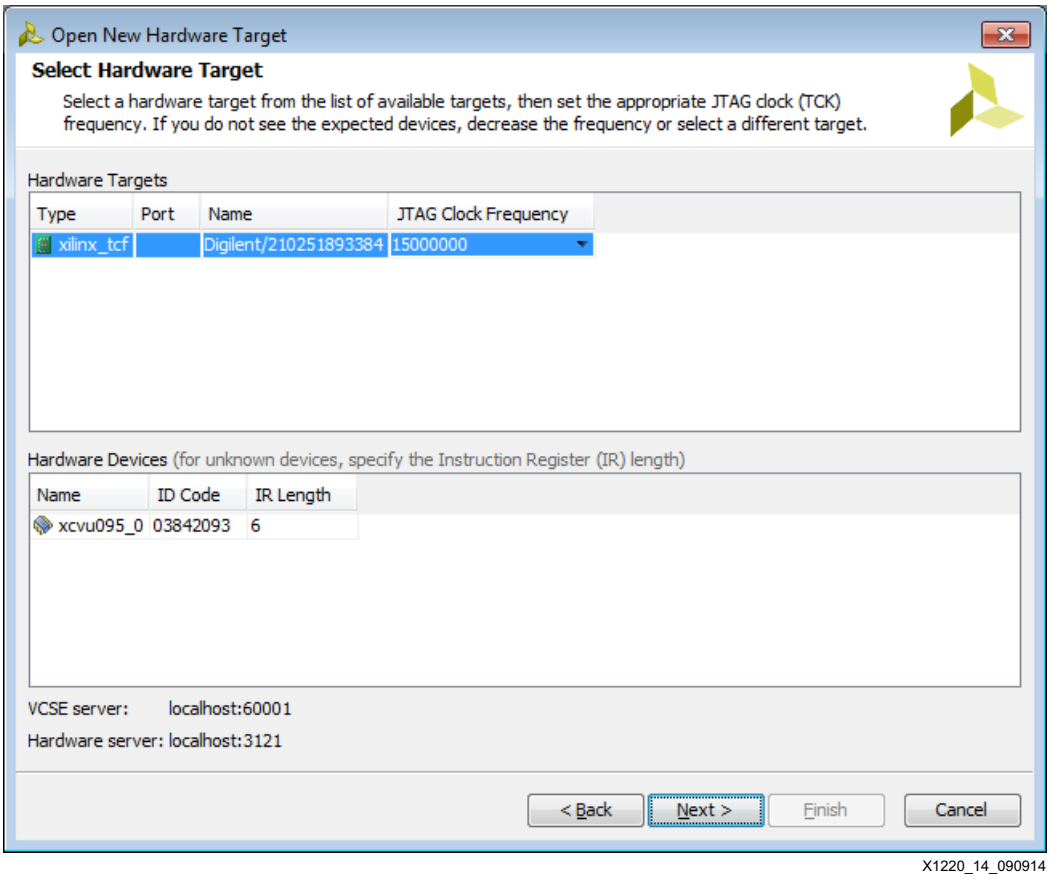

図 14: Open Hardware Target Wizard の JTAG [Select Hardware Target] ウィンドウ

4. Flow Navigator の [Program and Debug] の下にある [Hardware Manager] で、[Add Configuration Memory] → [XCVU095_0] をクリックします。図 15 に示すように、コンフィギュレーション メモリ パーツを選択するためのビューが表示されます。Micron (28F00AG18F) MT28GU01GAAX1E を選択して、[OK] をクリックします。

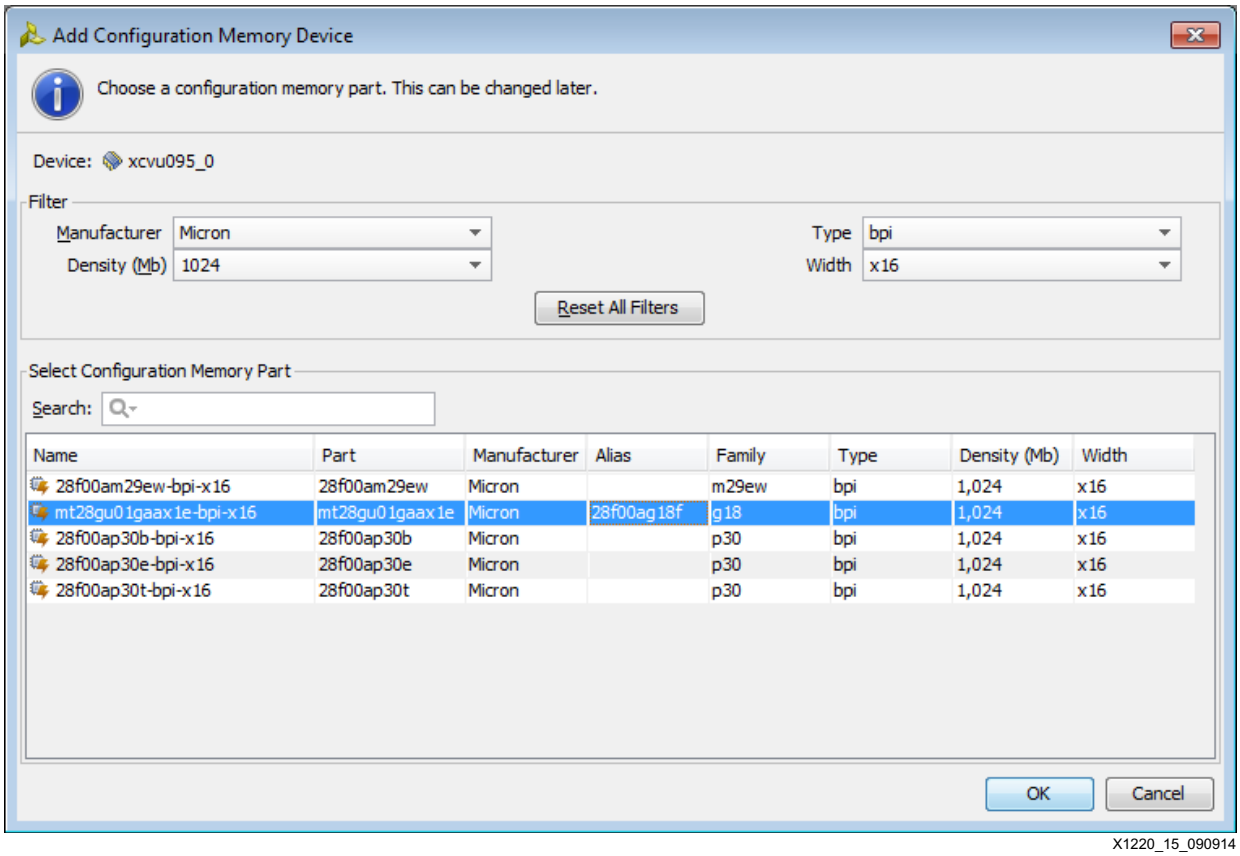

図 15: フラッシュ パーツの選択

5. デフォルトでは、コンフィギュレーション メモリ デバイスをプログラムするかどうか確認するダイアログ ボックスが表示されます。[OK] をクリックして、図 16 に示すように、表示されるビューにメモリ デバイスのコンフィギュレーション ファイルを入力します。[OK] をクリックして、フラッシュのプログラムを開始します。

注記: RS[1:0] がリビジョン管理に使用されている場合は、RS[1:0] に置き換えられた FPGA アドレス ピンを Vivado IDE に指示する必要があります。

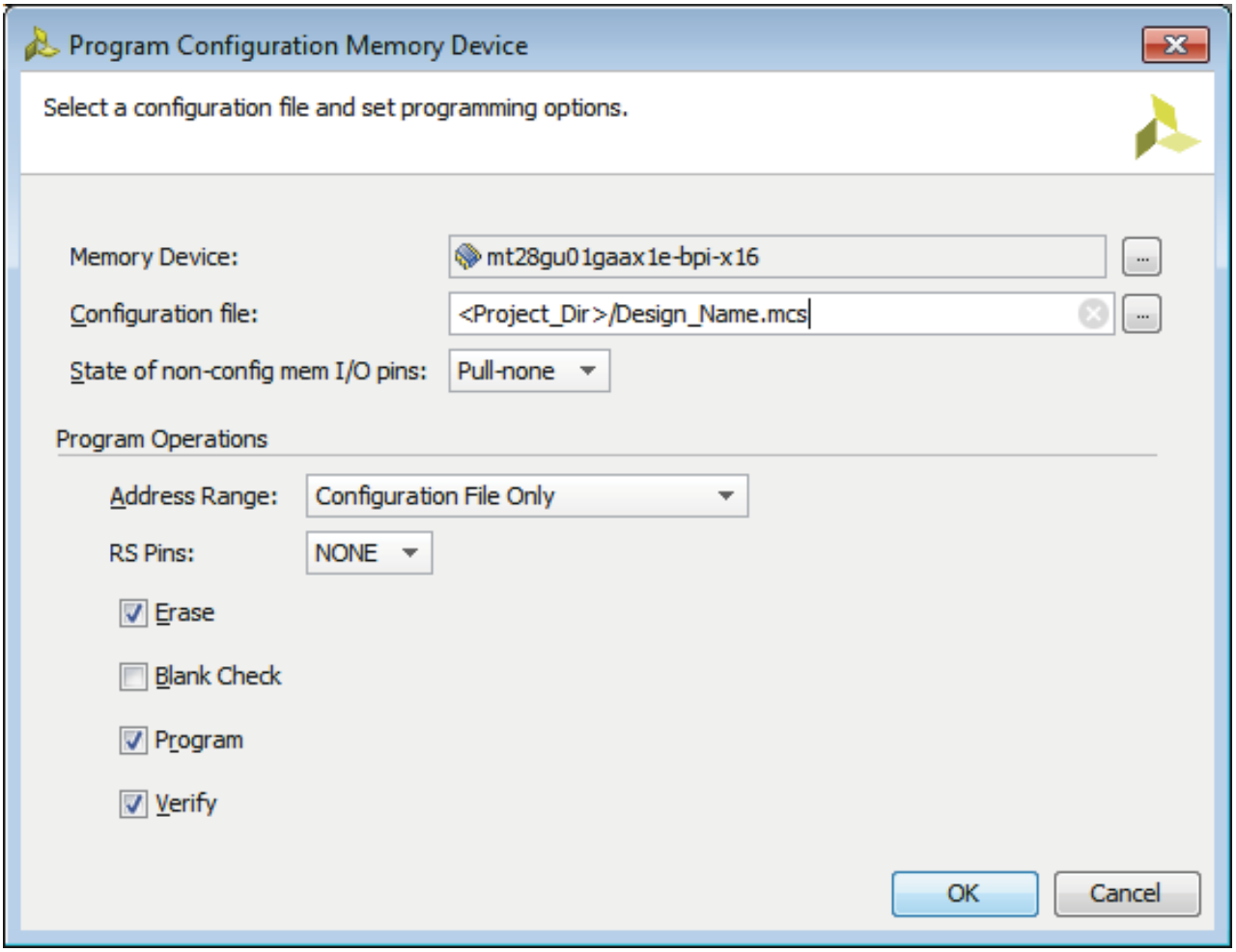

図 16: パラレル NOR フラッシュのプログラム

注記: ほとんどの場合、[State of non-config mem I/O pins] にはデフォルトの設定を使用します。間接プログラム中にユーザー I/O をすべてプルアップまたはプルダウンする必要があるデザインでは、ほかの設定を選択できます。

プログラミングが正常に完了した後、FPGA 上で PROGRAM_B 信号を Low にパルスするか、ボードの電源を切って入れ直します。これで FPGA は、同期読み出しを使用する BPI コンフィギュレーション モードで、パラレル NOR フラッシュからデザインをコンフィギュレーションします。詳細は、24 ページの「BPI コンフィギュレーションのシーケンス」を参照してください。詳細は、『FPGA および SoC 用 UltraFast 設計手法ガイド』(UG949) [参照 7] を参照してください。

プログラム時間

周波数 15 MHz で USB ケーブルを使用する XCVU095 ビットストリームについて、圧縮した場合と未圧縮の場合の動作時間の例を紹介します (表 2 参照)。これらの値は、参照用であり保証された値ではありません。

表 2: フラッシュ動作の参照用概算時間

イメージ	時間 (秒)		
	フラッシュの消去 (1)	フラッシュのプログラム	フラッシュの検証
圧縮された XCVU095 ビットストリーム (40,623,464 ビット)	28	53	13
非圧縮の XCVU095 ビットストリーム (286,746,912 ビット)	27	321	47
1 Gb フル イメージ (1,073,741,824 ビット)	222	1221	197

注記:

1. 消去時間は、格納されているフラッシュ データの量によって異なります。

Tcl コンソールによるパラレル NOR フラッシュの間接プログラム例

このセクションのコードは、19 ページの「Vivado IDE によるパラレル NOR フラッシュのプログラム例」の手順に従って実行される Tcl コマンドを示しています。

```
# Vivado script to program a parallel NOR flash
# The board should be connected to a programming cable and powered prior to running a script.
# The programming file is specified by the property PROGRAM_FILES in this example.
# Run this script from a Vivado command prompt: vivado -mode batch -source program_bpi.tcl

open_hw
connect_hw_server -url localhost:3121

# Set the current FPGA target. If multiple devices are in the JTAG chain, use the get_hw_devices
# command to help set the target FPGA.
# For setups with multiple cable connections, the user would have to select a specific target cable.
# Available cable frequencies are dependent on target cable. If a non-default frequency is desired
# this can be specified with the set_property PARAM.FREQUENCY. See UG908.

open_hw_target
current_hw_device [lindex [get_hw_devices] 0]

# Select target flash

create_hw_cfgmem -hw_device [lindex [get_hw_devices] 0] -mem_dev [lindex [get_cfgmem_parts
{mt28gu01gaax1e-bpi-x16}] 0]

# Set the address range for flash operations to the size of the programming file.
set_property PROGRAM.ADDRESS_RANGE {use_file} [ get_property PROGRAM.HW_CFGMEM [lindex
[get_hw_devices] 0 ]]

# Set the flash programming file
set_property PROGRAM.FILES {C:/Vivado_Workspace/Design/Design_Name.mcs} [ get_property
PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0]]

# Set the termination of unused pins when programming the flash
set_property PROGRAM.UNUSED_PIN_TERMINATION {pull-none} [ get_property PROGRAM.HW_CFGMEM [lindex
[get_hw_devices] 0 ]]
```

```
# Set the Revision Select pins, if unused the setting will be none.
set_property PROGRAM.BPI_RS_PINS {none} [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices]
0 ]]

# Set programming options for erase, program and verify. Blank check is not performed in this sample
# example.
set_property PROGRAM.BLANK_CHECK 0 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
set_property PROGRAM.ERASE 1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
set_property PROGRAM.CFG_PROGRAM 1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
set_property PROGRAM.VERIFY 1 [ get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]

# Check to ensure FPGA selected and supported flash memory are selected
startgroup
if {![string equal [get_property PROGRAM.HW_CFGMEM_TYPE [lindex [get_hw_devices] 0]] [get_property
MEM_TYPE [get_property CFGMEM_PART [get_property PROGRAM.HW_CFGMEM
[lindex [get_hw_devices] 0 ]]]]] } { create_hw_bitstream -hw_device [lindex [get_hw_devices] 0]
[get_property PROGRAM.HW_CFGMEM_BITFILE [ lindex [get_hw_devices]
0]]; program_hw_devices [lindex [get_hw_devices] 0]; };

# Program the flash
program_hw_cfgmem -hw_cfgmem [get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices] 0 ]]
refresh_hw_device [lindex [get_hw_devices] 0]
endgroup
```

BPI コンフィギュレーションのシーケンス

パラレル NOR フラッシュのプログラムが正常に完了した後、ボードの電源を切って入れ直すか、PROGRAM_B リセット信号をパルスすると、BPI コンフィギュレーション シーケンスが開始されます。このセクションでは、基本的なシーケンスについて簡潔に説明します。詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] を参照してください。

BPI コンフィギュレーション モード (M[2:0] = 010) では、UltraScale FPGA は、モード ピンをサンプルした後、INIT_B を初期化してリリースします。INIT_B 信号がリリースされ、インクリメントされた有効アドレス A[28:00] で制御信号 (FCS_B、FOE_B、ADV_B) がアサートされると、データ バス D[15:0] 上でフラッシュからデータがキャプチャされます。デフォルトの内部コンフィギュレーション クロック CCLK を使用して、ビットストリーム ヘッダーが読み出され、デザインで使用される読み出しモードが決定します。ビットストリーム ヘッダーに同期コマンドが含まれている場合は、FPGA コンフィギュレーション コントローラーが、接続されているフラッシュの RCR (読み出しコンフィギュレーション レジスタ) に対して非同期書き込みを実行し、同期モード ビットおよびレイテンシ ビットを設定します。フラッシュの RCR への書き込みが完了すると、FPGA コントローラーはビットストリーム ヘッダーの読み出しを開始します。ビットストリーム ヘッダーに基づいて、内部コンフィギュレーション クロックを使用するか、それともユーザーが選択したコンフィギュレーション クロック (すなわち、EMCCLK) に切り替えるかが決まります。EMCCLK が選択されている場合は、ビットストリーム データの同期読み出しが EMCCLK レートで実行されます。コンフィギュレーション完了後、フラッシュは同期読み出しモードを維持します。

図 17 のタイミング波形は、同期読み出しモードの BPI コンフィギュレーションを開始する手順を表しています。

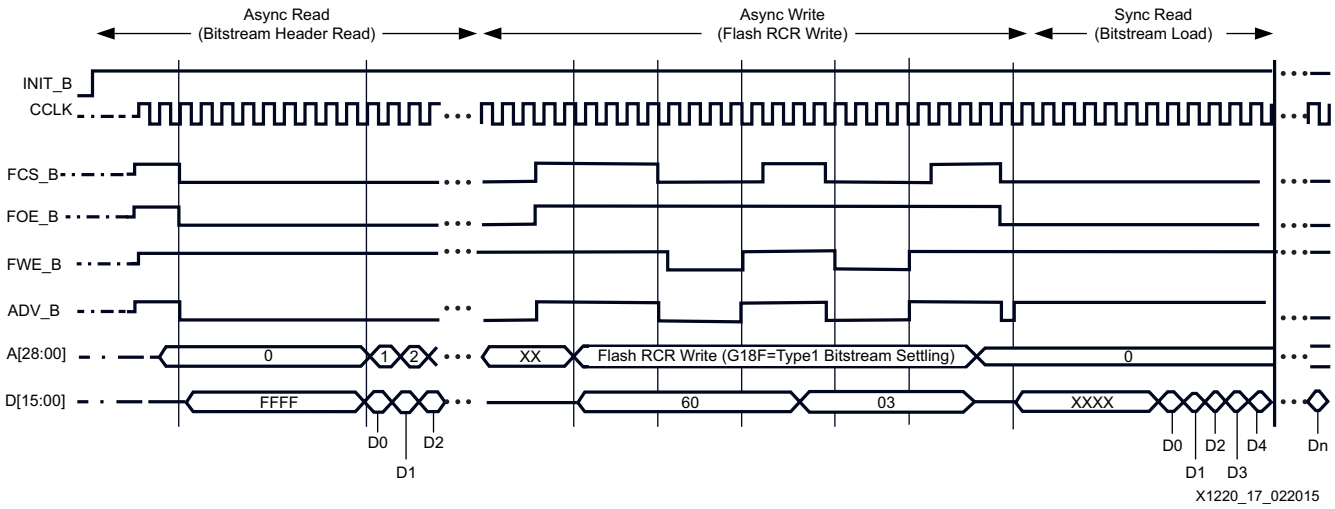

図 17: マスター BPI コンフィギュレーション モードの同期読み出し波形

コンフィギュレーション時間

このアプリケーション ノートで説明している BPI コンフィギュレーションのセットアップは、EMCCLK を使用します。EMCCLK の最大周波数を計算するには、パラレル NOR フラッシュの Clock-to-Out 仕様とデータシートに記載の FPGA セットアップ仕様の両方が必要です。サポートされる最大 EMCCLK 周波数 (F_{EMCCK}) は、UltraScale ファミリ データシート [参照 4]、[参照 5] に記載されており、これらの値を超えることはできません。BPI コンフィギュレーションの最大 EMCCLK の概算値は、式 1 から求められます。

$$Frequency_{MAX} = \frac{1}{a + b + c} \quad \text{式 1}$$

説明:

a = フラッシュの Clock-to-Out 値 (T_{CHQV})
b = FPGA のデータ セットアップ値 (T_{BPIDCC})
c = フラッシュの I/O ピンと FPGA の I/O ピンの間の伝搬遅延

たとえば、T_{CHQV} が 5.5 ns で T_{BPIDCC} が 3.5 ns の場合、ボード遅延が無視できるレベルであれば、最大周波数は 111 MHz になります。実際の T_{CHQV} および T_{BPIDCC} の仕様については、フラッシュのデータシートと FPGA のデータシートを参照してください。

表 3 では、90 MHz の EMCCLK を使用した場合の BPI コンフィギュレーション時間と、同じボード上で非同期読み出しおよび CCLK を使用した場合のコンフィギュレーション時間を比較しています。コンフィギュレーションの概算時間は、式 2 から求められます。パワーオン リセットからのコンフィギュレーション時間を計算する必要がある場合は、式 2 から得られた時間に、FPGA のデータシートに記載された T_{POR} の値を加算します。アプリケーションでパワーオン リセット時間が重視される場合は、表 1 を参照し、POR_OVERRIDE を使用して T_{POR} 時間を短縮することを検討してください。

$$ConfigurationTime = \frac{a}{b \times c} \quad \text{式 2}$$

説明:

a = ビットストリームのサイズ
b = コンフィギュレーション クロック周波数
c = データ バス幅

表 3 に、BPI コンフィギュレーション時間の例を示します。

表 3: Virtex UltraScale VU095 FPGA の BPI コンフィギュレーション時間の例

読み出しモード	コンフィギュレーション クロック ソース	フラッシュ データ幅	ビットストリームの コンフィギュレーション時間 (286,746,912 ビット)
同期読み出し (基準となる例)	EMCCLK (90 MHz オシレーター)	x16	199.13 ms
非同期読み出し	CCLK = 6 MHz	x16	3 秒
非同期読み出し	CCLK = 3 MHz (デフォルト設定)	x16	6 秒

非同期読み出しの計算では、追加のパラメーターを考慮に入れる必要があります。CCLK 偏差 ($F_{MCCKTOL}$) により、ビットストリーム周波数の設定は公称値 6 MHz に制限されます。

コンフィギュレーションのヒントについては、「デバッグ ガイダンス」を参照してください。

デバッグ ガイダンス

このセクションでは、BPI コンフィギュレーション モードおよびフラッシュの間接プログラムに関するチェックリストと一般的なデバッグ手順について簡潔に説明します。

回路図接続の検証

- UltraScale FPGA のデータ ピン D[03:00] はバンク 0 に割り当てられ、D[15:04] はバンク 65 の多目的ピンです。両方のピンが同じ電圧で動作していることを確認します。
- 1.8V の Micron 28F00AG18F フラッシュとの互換性のために、CFGBVS は GND に接続し、V_{CCO_0} および V_{CCO_65} には 1.8V 電源を供給します。
- JTAG ピンはバンク 0 に割り当てられ、このバンクの同じ要件に従う必要があります。Virtex UltraScale FPGA の場合、バンク 0 は 1.8V に設定されます。
- マスター BPI コンフィギュレーション モードの接続図を確認します。

  注記: FPGA のアドレス バスは A[25:0] ですが、Micron 28F00AG18F フラッシュのアドレス バスは A[26:1] です。バス ラベルが異なるために、FPGA とフラッシュ間の接続の大部分は 1 ずつずれているように見えます。
- x16 でサポートされるパラレル NOR フラッシュ ファミリには、LSB アドレス信号として A[1] を使用するものと A[0] を使用するものがあります。
- JTAG の TCK 信号と FPGA の CCLK 信号のシグナル インテグリティが維持されていることが非常に重要です。長い接続はできるだけ使用しないでください。接続が長いと、望ましくないノイズや電圧波形の反射が発生し、FPGA のシグナル インテグリティを低下させることがあります。詳細は、『UltraScale アーキテクチャ PCB デザイン ユーザー ガイド』(UG583) [参照 6] の「単方向のトポグラフィと終端」を参照してください。

適切なファイル生成

- MCS ファイルが適切に生成されていることを確認します。インターフェイスは BPIx16 に設定され、ビットストリームはアドレス `0x0` から上位のアドレスにロードされている必要があります。
- ビットストリームの設定で EMCCLK オプションが有効になっている場合は、I/O 規格が定義されていることを確認します。EMCCLK は多目的ピンであるため、デフォルトでは I/O 規格は未定義です。デザインに EMCCLK ピンを使用する場合、制約ファイルで EMCCLK の I/O 規格が適切に制約されていれば、それ以上の処理は必要ありません。EMCCLK ピンをコンフィギュレーションのクロッキングの目的でのみ使用する場合は、次の行を追加します。

```
set_property CONFIG_VOLTAGE 1.8 [current_design]
```

- BPI コンフィギュレーションでは、ターゲット フラッシュ デバイスに応じて同期モードのタイプ (Type1、Type2) が正しく設定されていることを確認します (28F00AG18F では Type1、P30 では Type2)。
- オプションで、コンフィギュレーションとプログラムをさらに高速化するには、ビットストリーム圧縮の設定を使用します。
- マスター CCLK に内部オシレーター ソースを使用する場合、ConfigRate オプションは、非同期読み出しモードでターゲット フラッシュおよび FPGA でサポートされる最大周波数を超えないようにします。FPGA の CCLK 偏差 $F_{MCCKTOL}$の仕様については、UltraScale データシート [参照 4] および [参照 5] を参照してください。
- ［Project Settings］(12 ページの図 7) のビットストリーム設定を適用する際は、デザインの合成が正常に完了し、合成済みのデザインが開かれていることを確認してください。Vivado Design Suite では、合成済みまたはインプリメント済みのデザインが開かれるまで、高度なビットストリーム設定は適用されません。
- Vivado Design Suite でビットストリーム イメージが見つからない場合は、フラッシュのプログラム ファイルのファイル パスにスペースが含まれていないことと、Windows を使用している場合にはファイル パスが 260 文字を超えていないことを確認してください。ファイル パスを含む Tcl コマンド ラインは、スペースを検出すると、フラッシュのプログラム ファイルを取得しようとします。

パラレル NOR フラッシュの間接プログラムに失敗した場合に実行する手順

- フラッシュ デバイスは不揮発性です。プログラムする前に、フラッシュが消去されていることを確認してください。
- ブランク チェックを実行して、前回の消去動作を確認します。
- Vivado ハードウェア マネージャーで、デザインに合った適切なフラッシュ メモリ パーツが選択されていることを確認します。完全なパーツ名については、フラッシュ メーカーのデータシートを参照してください。
- フラッシュ デバイスの動作不良が観察された場合は、ターゲット ケーブルの周波数を下げてください。これによって、ボードにおけるシグナル インテグリティの問題を切り分けやすくなります。
- JTAG チェーンのインテグリティが良好であることを確認します。
  - FPGA の IDCODE の基本動作を実行し、接続状態を確認します。ケーブルの接続時にターゲット ハードウェアが表示されない場合は、ケーブルの接続とボードの電源を確認してください。
  - FPGA 内に簡単なビットストリームをプログラムします。
  - フラッシュの間接プログラムの動作が正常に完了しない場合は、読み出された CFI (Common Flash Memory Interface) Manufacturer ID およびメモリ タイプが、ターゲット ファミリと一致することを確認してください。Vivado Design Suite は、実行するすべてのフラッシュ操作の最初の手順で CFI Manufacturer ID (Mfg ID) をキャプチャします。28F00AG18F の場合、Vivado Design Suite は正しくは次の情報を返します。

    Mfg ID : 89　Memory Type : 88b0　Device ID 1 : 0　Device ID 2: 0

    Vivado Design Suite からの結果がデータシートと一致しない場合は、適切なフラッシュ ファミリおよび集積度を選択していることを確認してください。
  - さらなる分析のために FPGA ステータス レジスタをキャプチャします。

BPI コンフィギュレーションに失敗した場合に実行する手順

- 電源を切って入れ直してもコンフィギュレーションが成功しなかった場合は、FPGA とフラッシュ デバイス間の電源投入シーケンスに問題がないように、PROGRAM_B 信号をパルスします。
- 適切な時間内にコンフィギュレーションが完了しない場合は、write_bitstream が、使用する読み出しモードおよびコンフィギュレーション クロックに合わせて設定されていることを確認します。デフォルトの非同期読み出しおよびデフォルトの内部コンフィギュレーション クロックの速度は、ビットストリームの同期読み出しおよび EMCCLK オプションを有効にした場合よりもはるかに低速です。
- 内部 CCLK およびデフォルトのビットストリーム設定を使用して、まず基本的な非同期ビットストリームを試します。これによって、シグナル インテグリティの問題とビットストリーム プロパティの問題を切り分けることができます。デフォルトのコンフィギュレーション クロック プロパティ (3 MHz) を使用して FPGA をコンフィギュレーションすると、同期読み出しと EMCCLK を有効にしてビットストリームを読み出す場合に比べて、所要時間が大幅に長くなります。コンフィギュレーション時間については、表 3 を参照してください。
- 「適切なファイル生成」に従って、MCS ファイルを確認します。
- Vivado デバイス プログラマを使用して、コンフィギュレーション ステータス レジスタを確認します。
- Vivado デバイス リードバックを使用して、適切なビットストリーム パターンがフラッシュ メモリに格納されていることを確認します。

参考資料

1. 『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570)
2. 『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908)
3. Micron StrataFlash Embedded Memory MT28GU01GAAA1E (28F00AG18F) データシート
4. 『Kintex UltraScale FPGA データシート: DC 特性および AC スイッチ特性』(DS892)
5. 『Virtex UltraScale FPGA データシート: DC 特性および AC スイッチ特性』(DS893)
6. 『UltraScale アーキテクチャ PCB デザイン ユーザー ガイド』(UG583)
7. 『FPGA および SoC 用 UltraFast 設計手法ガイド』(UG949)

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2025 年 5 月 28 日	1.2.1	編集上の更新のみ。技術的内容の更新なし。
2022 年 3 月 16 日	1.2	「概要」に、同期読み出しを使用するマスター BPI コンフィギュレーション モードが新規デザインに推奨されないことに関する注記を追加。
2015 年 3 月 18 日	1.1	図 17 を更新。POR_OVERRIDE および EMCCLK の使用に関する詳細を追加。
2014 年 12 月 8 日	1.0	初版

お読みください: 重要な法的通知

本資料に掲載されている情報は情報提供のみを目的としており、技術的に不正確な情報、省略、誤植を含む可能性があります。ここに記載の情報は、製品およびロードマップの変更、コンポーネントとマザーボードのバージョン変更、新しいモデルや製品のリリース、製造業者の違いによる製品の差異、ソフトウェア変更、BIOS のフラッシュ、ファームウェア アップグレードなどを含む (ただしこれに限定されない) 多くの理由によって変更される場合があり、不正確となる可能性があります。いかなるコンピューター システムにも完全に防止または軽減できないセキュリティの脆弱性のリスクが存在します。AMD は、情報の更新、または修正や改訂について一切の義務を負わないものとします。ただし AMD は本情報を改訂してその内容を随時変更する権利を有し、その改訂または変更に関して通知する義務は負いません。本情報は「現状有姿」で提供されます。AMD は本書の内容について一切の表明も保証も行わないものとし、本情報の不正確さ、誤り、省略について一切の責任を負いません。また、AMD は非侵害性、商品性、または特定目的適合性に関する黙示的保証を行わないものとします。AMD は本資料に含まれる情報の使用に起因する依存、直接、間接、特別、または結果的な損害に対して、AMD が当該の損害の可能性を知らされていた場合であっても、責任を負うことはありません。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとします。セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品責任の制限を規定する適用法令および規則にのみ従うものとします。

© Copyright 2014–2025 Advanced Micro Devices、Inc. AMD、AMD の矢印形のロゴ、Kintex、Spartan、UltraScale、UltraScale+、Virtex、Vivado、およびその組み合わせは Advanced Micro Devices, Inc の商標です。本書に使用されているその他の製品名は識別目的のみに使用されており、所有するそれぞれの企業の商標である可能性があります。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@amd.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。