XAPP1328 (v1.0) 2018 年 3 月 5 日

NOR フラッシュを使用するコンフィギュレーションおよびブートで部品調達の柔軟性とコスト削減を実現

著者: Eric Crabill

概要

ザイリンクス デバイスは、コンフィギュレーションやブート用に市販の NOR フラッシュを使用できます。この汎用性は部品調達の柔軟性とコスト削減に役立ちますが、システム設計者は代替設計フローに従って NOR フラッシュを使用する必要があります。このアプリケーション ノートで提供するコンフィギュレーション ソリューションまたはブート ソリューションの推奨事項では、NOR フラッシュを使用する場合の一般的な課題およびコスト削減の可能性について説明します。

はじめに

ザイリンクス デバイスは SRAM ベースであるため、電源投入後に少なくとも 1 回はプログラミング情報を読み込む必要があります。その読み込み方法は、プログラミング情報が読み込まれるまで待機する受動的方法と、デバイスが情報を自動的に取り込む能動的方法などさまざまです。最も一般的な能動的方法は、NOR フラッシュをザイリンクス デバイスに接続し、ザイリンクス デバイスが自動的にコンフィギュレーションまたはブートできるようにプログラミング情報をこの不揮発デバイスに格納する方法です。

コンフィギュレーションやブートに NOR フラッシュを使用するザイリンクス デバイス ベースのシステムでは、NOR フラッシュを選択する際に注意すべき点が多数あります。たとえば、サイズ制限があるシステムの場合、ボード エリアを最小に抑えるために厳しいパッケージ要件の NOR フラッシュを選択する必要があります。同様に、電源投入時のレイテンシに制限があるシステム デザインでは、コンフィギュレーションやブート時間をできるだけ短くするために広帯域幅の NOR フラッシュを選択する必要があります。このような要件を満たした場合、選択したフラッシュはシステム デザイン要件を満たすことができますが、NOR フラッシュの調達の柔軟性やコストに大きな影響を及ぼす可能性があります。

NOR フラッシュとザイリンクス デバイスを使用するほとんどのシステム デザインは、コンフィギュレーション ソリューションやブート ソリューションに影響を与える制約がありません。ザイリンクスでは、制約のないシステム デザイン、長寿命を想定したシステム デザイン、コスト重視のシステム デザインの実現に向けて、NOR フラッシュを使用する代替設計案を推奨しています。この代替設計案の目的は次のとおりです。

- 部品調達の柔軟性を高める (第 1 の目的)
- コスト削減 (第 2 の目的)

この代替設計案がこれらの目的達成に役立つことを理解するために、関連する要素について簡単に説明します。

関連する要素

システムの寿命

システムが製造される期間はどれくらいか、製造終了後に展開されたシステムの拡張メンテナンスは必要か、などを考えます。製造や拡張メンテナンスで永続的な複数 NOR フラッシュの置き換えに対応する必要がある航空機の飛行制御装置を例として考えます。

システム コスト

NOR フラッシュの価格がプロジェクトの利益率に大きく影響する場合、システム コストを優先するのか、戦略的あるいはタイミングを利用したコスト削減を優先するのかを考えます。市販のエンターテインメント端末では、購入時の最低価格に基づいて、NOR フラッシュの形態、適合性、および機能代替の柔軟性に伴うコスト削減が必要になる可能性があります。

インターフェイス

NOR フラッシュの選択肢に影響を与えるインターフェイスのトレンドがあるかを考えます。これまではトレンドというようなものはありませんでしたが、NOR フラッシュの市場データに基づく限りでは、シリアル インターフェイスの方がパラレル インターフェイスよりも好まれる傾向があります。その結果、シリアル NOR フラッシュは、パラレル NOR フラッシュよりも多くのベンダーで提供され、供給期間が長い傾向が見られます。

供給の制約

システムの寿命期間において調達の柔軟性に対する要求が高まる可能性がある一方で、供給の制約リスクが現時点または将来的にあるかを考えます。NOR フラッシュ、NAND フラッシュ、SRAM および DRAM は、市場の力によって供給や価格が大きく変動します。調達の柔軟性を備えることによって、このようなリスクに対応できます。

製造中止またはベンダーの撤退

使用している NOR フラッシュの製造が中止される可能性がある、または該当する NOR フラッシュ ベンダーが市場を撤退する可能性が現時点または将来的にあるかを考えます。NOR フラッシュ ベンダーは十分な製品寿命を提供しますが、これはベンダーの独立性と経営状態が良いことが前提です。

代替設計

NOR フラッシュを使用する代替設計案の適用では、システムの設計と製造において一部に変更を加える必要があります。次のセクションでは、これらの部分について説明し、代替設計の推奨方法を示します。システム設計者は、これらの推奨事項に基づいてそれぞれのリスク トレランスと影響を与える要素を確認し、その自由度の範囲内で調達の柔軟性を最大限に高めながらコストを削減できる最適な判断をします。

プロビジョニング - ボードおよび NOR フラッシュのプログラミング

NOR フラッシュをベースとするコンフィギュレーションまたはブート ソリューションの場合、プロビジョニング プロセスには、プリント回路基板上に NOR フラッシュを配置し、フラッシュをプログラムするために必要なコネクティビティを提供することが含まれます。これらを計画する前に、シリアル インターフェイスまたはパラレル インターフェイスの選択という重要な判断をする必要があります。

市場ではパラレル NOR フラッシュが徐々に減少していることを考慮して判断してください。パラレル NOR フラッシュの使用を推奨するシステムがありますが、ザイリンクスでは、代替ソリューションを提供しており、パラレル NOR フラッシュを使用する必要性はなくなりました。

- 統合型 PCI Express® (PCIe) ペリフェラルを使用すると、オープン システムで使用される PCIe インターフェイスを備えたアプリケーションでは、多くの Zynq® UltraScale+™ MPSoC および Zynq-7000 SoC の起動にシリアル NOR フラッシュを利用できます。
- Tandem コンフィギュレーションを使用すると、オープン システムで使用される PCIe インターフェイスを備えたアプリケーションで UltraScale™、UltraScale+、および 7 シリーズ FPGA のコンフィギュレーションにシリアル NOR フラッシュを利用できます。
- UltraScale および UltraScale+ FPGA で有効なデュアル Quad-SPI コンフィギュレーション手法は、新しいデザインで使用できるパラレル NOR フラッシュよりも広い帯域幅を実現します。

NOR フラッシュ市場のインターフェイス動向とザイリンクスの代替ソリューションを考えると、パラレル NOR フラッシュは単一ソースのコンポーネントとして見なすことが妥当であり、代替設計案を使用するアプローチには適切ではありません。通常、単一ソースのコンポーネントは、システム デザインに優れた価値や差別化をもたらします。ほかの懸念事項をカバーするほどのメリットをシステム デザインにもたらすこともありますが、複数ソースの汎用コンポーネントとは異なる計画が必要です。つまり、コンポーネントの有効期間を把握し、最終購入の機会や在庫管理など生産/販売終了オプションについてベンダーと協議する必要があります。

このような考慮事項があるため、パラレル NOR フラッシュを使用する明確かつ説得力のある理由がない限り、ザイリンクスはザイリンクス デバイスのコンフィギュレーションやブートにはシリアル NOR フラッシュを使用することを推奨しています。代替設計フローでは、シリアル NOR フラッシュを選択する必要があり、このアプリケーション ノートで示す推奨事項ではシリアル NOR フラッシュの使用を前提としています。

シリアル NOR フラッシュのパッケージと PCB フットプリント デザイン

複数のシリアル NOR フラッシュ ベンダーがさまざまなパッケージ オプションを提供しています。SO16W、SO8W、BGA24 などの複数ベンダーで提供されているパッケージ オプションを使用してください。幸いにも、ベンダーはこれらの共通パッケージにおいてピンの互換性を重視しています。意図的に独占的な供給状況を作るために、WLCSP などの独自またはベンダー固有のパッケージ オプションを使用することは控えてください。

柔軟性を高めるために、少なくとも 3 種類のパッケージに対応できるシリアル NOR フラッシュの PCB フットプリントを実装することが可能です。複数のベンダー間におけるパッケージ オプションと NOR フラッシュ メモリ容量の比較結果に基づいて、互換性のあるパッケージで 2 つの可能性があります。

- 64Mb およびそれ以下の場合: SO16W (300 mil)、SO8W (208 mil)、MLP8 (6×5 mm)
- 128Mb およびそれ以上の場合: SO16W (300 mil)、BGA24 (5×5)、BGA24 (4×6)

図 1 は 64Mb 以下の場合を示しており、複合の MLP8 (6×5mm) と SO8W (208mil) ランドが大規模な SO16W (300mil) ランド内に適合しています。スペースがコスト高を招く場合は、柔軟性が低下しますが SO16W を省略できます。

図 2 は 128Mb 以上の場合を示しており、複合の BGA24 (4×6) と BGA24 (5×5) ランドが大規模な SO16W (300mil) ランド内

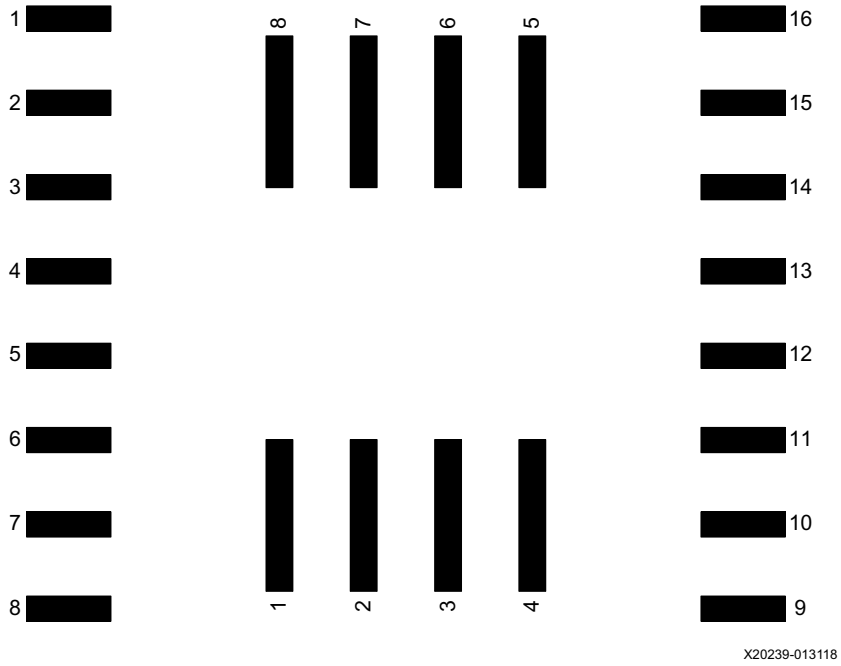

図 1: 複合の MLP8 (6x5mm)、SO8W (208 mil)、および SO16W (300mil)

に適合しています。小規模の場合と同様、SO16W は省略できます。これらの BGA24 パッケージは同じピン配置を備えているため、いずれかを 5×6 アレイに配置できます。

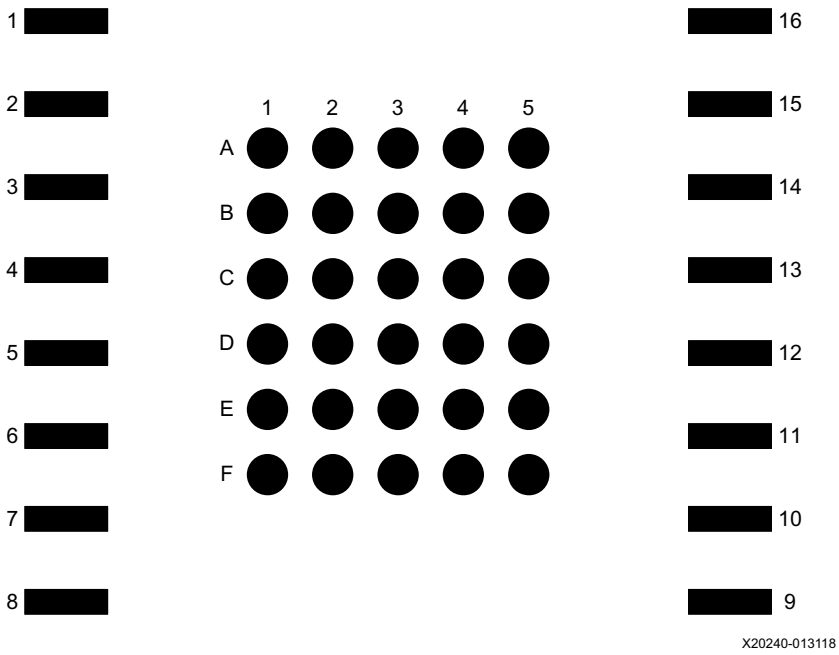

図 2: 複合の BGA24 (4x6)、 BGA24 (5x5)、 および SO16W (300mil)

シリアル NOR フラッシュは広範な電圧に対応できます。電源はシリアル NOR フラッシュの I/O やコアの電力供給に重要となるため、特に注意が必要です。ほとんどのシリアル NOR フラッシュは、I/O とコアの電力供給用として共有される V_{CC} を 1 つ使用し、I/O のスイッチングしきい値に基づいて必要な電圧 (3.3V または 1.8V) を使用します。ただし、一部のシリアル NOR フラッシュでは、コアと I/O の電力供給用として V_{CC} と V_{IO} をそれぞれに使用します。個別に V_{IO} を必要とするデバイスの場合は、通常、別のシリアル NOR フラッシュの N/C ピンを使用します。

フットプリントが単一または複合のいずれであっても、Cypress Semiconductor S25FL-S ファミリの特定デバイスのように個別の V_{IO} を備えたデバイスを使用できるようにするために、ボード設計で V_{IO} の接続を分割して柔軟性を与える必要があります。これは、0Ω 抵抗を介して V_{IO} のランドを任意の V_{IO} 供給レールに接続することで簡単にできます。V_{IO} を必要とするシリアル NOR フラッシュの場合はボードにこの抵抗を組み込む必要がありますが、それ以外の場合には省略します。

シリアル NOR フラッシュには、クロック、セレクト、データ ラインなどの基本的な機能のほかにさまざまなピン機能 (ホールド、書き込み禁止、リセットなど) があります。多くのベンダーは注文コードを提供しており、ユーザーは購入したシリアル NOR フラッシュ上の補足機能を制御できます。一般的に、ホールドおよび書き込み禁止機能はアクティブ Low で、2 つの上位データ ピンで共有されます。これは、これらの信号にボード レベルのプルアップ抵抗が必要になる理由の 1 つとなります。

リセット機能は通常アクティブ Low です。Zynq UltraScale+ MPSoC、UltraScale/UltraScale+ FPGA、Zynq-7000 SoC、および 7 シリーズ FPGA の場合、シリアル NOR フラッシュが提供する専用リセット機能はザイリンクス デバイスに接続せずに、代わりにボード レベルのプルアップ抵抗へ接続します。この抵抗は、シリアル NOR フラッシュの必要性に応じて、ボードに組み込むことができます。

プログラミング方法の概要

シリアル NOR フラッシュをプログラムする方法は多数あります。開発、デバッグ、プロトタイピング、および検証では、ソケットやボード再設計の必要性を最小限にするためにインシステム プログラミング機能が必須になっています。製造期間中には、特定タイプのインシステム プログラミングを使用できますが、ほとんどの製造過程ではあらかじめプログラムしたシリアル NOR フラッシュを使用して、ボード上に直接組み立てます。

図 3 に、シリアル NOR フラッシュのインシステム プログラミングをサポートするコンセプトと接続を示します。この図は、ザイリンクス デバイスを経由する間接プログラミングとザイリンクス デバイスをバイパスする直接プログラミングの両方を表しています。

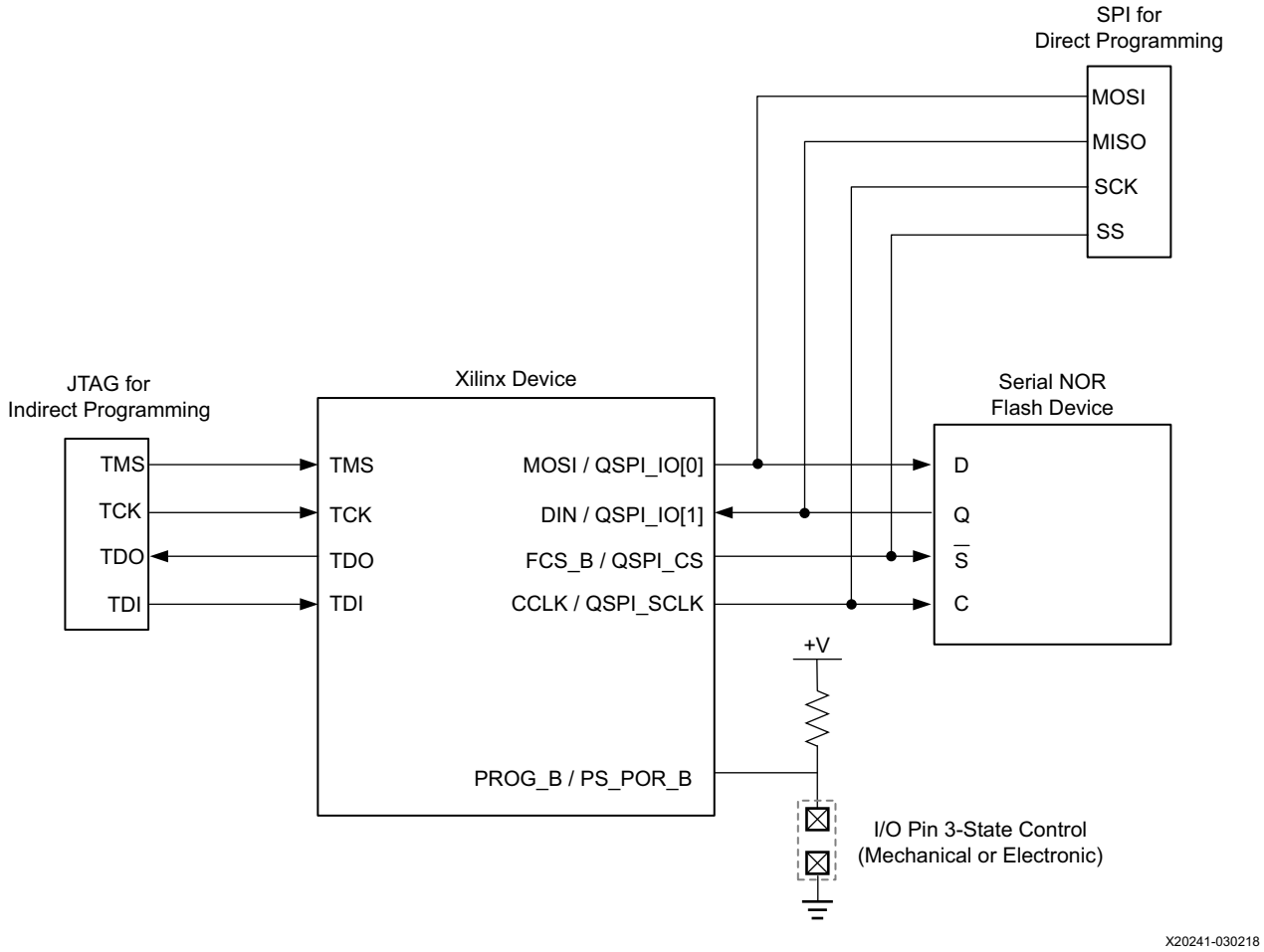

図 3: ザイリンクス デバイスの間接インシステム プログラミングと直接インシステム プログラミング

調達の柔軟性が向上し、コストも削減できるため、直接プログラミングと間接プログラミングの両方に対応できることが理想的です。いかなる場合でも、ボードレベルのシグナル インテグリティが重要です。システム設計者とボード設計者が連携して、クロック (SCK、TCK) 配線トポロジ、スタブの最小化、および SPI ライン上のさまざまなドライバーに特に注意しながら、JTAG と SPI トレースを慎重に配線する必要があります。IBIS を使用したボードレベルのモデリングを推奨しています。抽出した信号遅延は、制限範囲を超えた動作が実行されないようにするために JTAG および SPI インターフェイスのタイミング解析に役立ちます。

間接プログラミング - ザイリンクス デバイスを使用 (経由する)

間接プログラミングの主な方法は、図 3 に示すように JTAG ケーブルを接続して、Vivado® Design Suite で JTAG インターフェイスを制御します。Vivado Design Suite を使用してザイリンクス デバイスを設定または起動し、JTAG-SPI 間のブリッジを実装します。その後、ザイリンクス デバイスを介してシリアル NOR フラッシュにプログラミング データを送信します。

このブリッジは、ザイリンクス デバイスに応じて、ハードウェア、ソフトウェア、あるいは両方の組み合わせで実装できます。UltraScale、UltraScale+、および 7 シリーズ FPGA の場合は、独自ブリッジがプログラマブル ロジックにダウンロードされます。ザイリンクスでは、各 FPGA ファミリに対応するブリッジの実装をプリコンパイルおよび検証し、それらを Vivado Design Suite 内で管理しています。Zynq UltraScale+ MPSoC および Zynq-7000 AP SoC の場合は、U-Boot ベースのブリッジがプロセッシング システムにダウンロードされます。U-Boot は、オープン ソースの汎用ブートローダーであり、エンベデッド開発では頻繁に使用されます。

この間接プログラミング方法では、JTAG ケーブルと Vivado Design Suite のみ必要です。ほとんどのシステム デザインには、テスト/デバッグ用に JTAG が実装されているため、ボード コストは増加しません。開発、デバッグ、プロトタイプ、検証における使い易さがその最大のメリットです。Vivado Design Suite は、汎用のデバイス プログラマではないことに留意してください。Vivado Design Suite でプログラムできる最新のデバイス 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 1] を参照してください。また、量産向けプログラミングでの Vivado Design Suite の使用に関する重要事項は、ザイリンクス エンドユーザー使用許諾契約 (EULA) [参照 2] のセクション 4(a) を参照してください。

間接プログラミングのもう一つの方法は、フィールド アップデート可能なデザインを構築することです。まず、図 3 のように JTAG ケーブルを使用してデザインをプログラムしますが、デザインにはプログラミング情報を受信するための 2 つ目のインターフェイスがあります。フィールド アップデートの実行が促されると、デザインはこのインターフェイスを介してシリアル NOR フラッシュのプログラミング データを受信し、その後シリアル NOR フラッシュに書き込みが実行されます。

この間接プログラミング方法の場合は、JTAG ケーブルと Vivado Design Suite のほかに、選択したシリアル NOR フラッシュのフィールド アップデートをサポートできる十分な機能が実装されたシステム デザインが必要です。量産向けプログラミングには、システム ジャンプスタートの使用が最適です。

直接プログラミング - ザイリンクス デバイスをバイパス (回避する)

直接プログラミングは、図 3 に示すように SPI バスに接続されたサードパーティのインシステム プログラマを使用します。DediProg 社 [参照 3]、TotalPhase 社 [参照 4]、Corelis 社 [参照 5] などのベンダーから、さまざまなツールが提供されています。インシステム プログラマは USB で PC に接続されます。PC でソフトウェア アプリケーションを実行するため、ザイリンクス デバイスを介さずにシリアル NOR フラッシュへ直接プログラミング データを転送できます。つまり、ザイリンクス デバイスからの干渉がないと考えることもできます。

直接プログラミング中にザイリンクス デバイスからの干渉を防ぐ最も簡単な方法は、ザイリンクス デバイスをリセット状態に保持し、SPI バスへの I/O 接続を強制的にトライステートにすることです。図 3 に、機械的または電気的制御によるこの機能の形態を示しています。詳細は表 1 を参照してください。

表 1: SPI バスへのザイリンクス デバイスの I/O 接続を強制的にトライステートにする

ザイリンクス デバイス ファミリ	I/O ピンのトライステート制御	詳細
Zynq UltraScale+ MPSoC	PS_POR_B	電源投入時あるいはその後の任意の期間 Low を保持 (直接プログラミングの期間)。
UltraScale および UltraScale+ FPGA	PROGRAM_B	電源投入時あるいはその後の任意の期間 Low を保持 (直接プログラミングの期間)。
Zynq-7000 AP SoC	PS_POR_B	電源投入時あるいはその後の任意の期間 Low を保持 (直接プログラミングの期間)。
7 シリーズ FPGA	PROGRAM_B	電源投入後 Low を保持 (直接プログラミングの期間)。

図 3 では、サードパーティ インシステム プログラマ用に従来コネクタを推奨していますが、その他の接続方法も可能です。ボードにはんだ付けされた SO8W や SO16W パッケージにシリアル NOR フラッシュを含めることができるチップクリップを利用できます。Pomona Electronics 社 [参照 6] や 3M 社 [参照 7] などのベンダーがこのようなクリップを製造しています。プログラムされるユニットがわずかしかない場合は、フライング ワイヤをボードにはんだ付けできますが、この目的のために計画的に配置されたビアが信号トレース上に存在することが条件です。

直接プログラミングの場合は、サードパーティのインシステム プログラマが必要で、それを接続するためのボードレベルの準備、さらにザイリンクス デバイスをリセット状態に保持する機能も必要になります。ボード コストの増加はごくわずかです。サードパーティのインシステムプログラマは、非常に有益なツール価値を提供する割にわずかな投資ですみます。ツール価値とは、ユニバーサルなデバイス プログラマであること、そしてリアル NOR フラッシュの置き換えオプションが豊富にあることです。直接プログラミングの使用は、量産向けプログラミングに最適です。

推奨事項のまとめ

- 3 つのパッケージに対応できるように複合フットプリントを使用する。
- 直接プログラミングと間接プログラミングの両方に対応できるようにする。
  - Vivado Design Suite を使用する間接プログラミングは、『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 1] に記載されているシリアル NOR フラッシュを使用する簡単な方法を提供。
  - サードパーティのインシステム プログラマを使用する直接プログラミングには、代替として豊富なシリアル NOR フラッシュがある。
- IBIS を使用してボードレベルのモデリングを実行する。
  - JTAG および SPI インターフェイスのシグナル インテグリティを確保して設計する。
  - タイミング遅延を取得してタイミング解析ツールに提供する。

選択 - NOR フラッシュに関する技術的な考慮事項

ザイリンクス デバイスは、さまざまなシリアル NOR フラッシュに適合できるように設計されています。Vivado Design Suite でプログラムできるデバイスの一覧は、『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 1] を参照してください。その他にも多くのシリアル NOR フラッシュを使用できます。次に示す技術的な考慮事項を参照して、ザイリンクス デバイスと希望するシリアル NOR フラッシュの互換性、およびシステムの設計要件を確認してください。

容量と I/O 規格

候補となるシリアル NOR フラッシュは、ザイリンクス デバイス用の圧縮されていないプログラミング データ ペイロードを少なくとも 1 つ格納できる十分な容量が必要です。確実なフィールド アップデートをサポートするには、フィールド アップデート中の電源障害やその他の中断リスクに備えて、フォールバック用のゴールデン イメージを格納できるように 2 倍の容量が必要です。また、ザイリンクス デバイス用のプログラミング データ ペイロードのほかに、コードやデータのストレージ、OS イメージ、ファイル システムなどのアプリケーションに必要なものを格納するために、さらに容量が必要です。

使用するシリアル NOR フラッシュは、ザイリンクス デバイスのコンフィギュレーション インターフェイスやブート インターフェイスと互換性のある I/O 規格を備えている必要があります。通常、これはシステム デザイン要件やザイリンクス デバイスによって制約されます。コンフィギュレーション/ブート インターフェイスの一般的な I/O 規格は 3.3V または 1.8V です。すべてのデバイスが各データシートに記載されている仕様範囲内で動作する限りは、その他の電圧も使用可能です。

コマンド サポート

使用するシリアル NOR フラッシュは、コンフィギュレーションやブート プロセスの一貫としてザイリンクス デバイスで発行されるコマンドをサポートしている必要があります。幅広く互換性を持たせるために、ザイリンクス デバイスは最も一般的なコマンドを含む最小のコマンド セットを使用します。

UltraScale、UltraScale+、および 7 シリーズ FPGA がコンフィギュレーション時に発行できるコマンドを表 2 に示します。多くのコマンドはオプションとなります。

表 2: UltraScale、UltraScale+、および 7 シリーズ FPGA のコンフィギュレーション コマンド セット

コマンド コード	コマンド名	詳細
0x0B	Fast read、24 ビット アドレス、シングル出力	このコマンドは、コンフィギュレーション開始時に無条件に発行されるため、必須です。
0x3B	Fast read、24 ビット アドレス、デュアル出力	これらのコマンドは、プログラミング データが 128Mb を超えない場合、より広い帯域幅に適しています。これらはオプションであり、FPGA がこれらコマンドのいずれかを使用して残りのプログラミング データを読み取ろうとしていることをプログラミング データのヘッダーが示す場合にのみ発行されます。
0x6B	Fast read、24 ビット アドレス、クワッド出力	
0x0C	Fast read、32 ビット アドレス、シングル出力	これらのコマンドは、プログラミング データが 128Mb を超えない場合のオプションとなります。プログラミング データが 128Mb を超える場合は、これらコマンドのいずれか 1 つが必要になります。これらのコマンドは、FPGA がこれらのいずれかを使用して残りのプログラミング データを読み取ろうとしていることをプログラミング データのヘッダーが示す場合にのみ発行されます。
0x3C	Fast read、32 ビット アドレス、デュアル出力	
0x6C	Fast read、32 ビット アドレス、クワッド出力	

UltraScale、UltraScale+、および 7 シリーズ FPGA は、コンフィギュレーション中にネゴシエートしません。バス幅とコマンドは、プログラミング データが生成されるときにシステム設計者の指示に従ってセットされます。表 2 によると、128Mb を超えないデータをプログラムする場合、候補となるシリアル NOR フラッシュがサポートする必要がある最小コマンドは 0x0B です。一方、128Mb を超えるデータをプログラムする場合の最小コマンドは 0x0B と 0x0C です。

FPGA は、32 ビット アドレスを有効にするコマンドやクワッドモードを有効にするコマンドを発行しません。候補となるシリアル NOR フラッシュは、コンフィギュレーション中に FPGA が発行するコマンドに対応できるように、電源投入時には準備が整っている必要があります。これらの機能を恒久的に有効にするには、不揮発性制御レジスタの制御ビットをプログラムする必要があります。

さらに、FPGA によってプログラミング データ全体のペイロードが読み込まれる場合、最大で 2 つの読み出しコマンドが使用 (場合によっては 1 つのみ) されることを理解しておく必要があります。このため、候補となるシリアル NOR フラッシュは、単一の読み出しコマンドでフル アレイの読み出しをサポートする必要があります。つまり、1 つのリード コマンドを発行するだけで、すべてのデータを読み取ることができる必要があります。ほとんどのシリアル NOR フラッシュはフル アレイの読み出しをサポートしていますが、一部の製品はダイスタッキング技術を使用して構築されていたり、複数のチップ セレクトを備えている場合があり、フル アレイの読み出しをサポートしていない可能性があります。したがって、これらの確認が必要です。

Zynq UltraScale+ MPSoC には、シリアル NOR フラッシュ用に 2 つのブート モードがあります。モード ピンをブートストラップして、24 または 32 ビット アドレスを選択します。24 ビット アドレスが選択された場合、bootROM は表 3 に示すコマンドのみを発行できます。同様に、32 ビット アドレスが選択された場合、bootROM は表 4 に示すコマンドのみを発行できます。32 ビット アドレスの使用は、プログラミング データが 128Mb を超えない場合のオプションとなります。プログラミング データが 128Mb を超える場合は 32 ビット アドレスが必要です。次の表に示すコマンドはオプションです。

表 3: Zynq UltraScale+ MPSOC および Zynq-7000 SoC のブート コマンド セット、24 ビット アドレス

コマンド コード	コマンド名	詳細
0x03	Read、24 ビット アドレス、シングル出力	このコマンドは、ブート ヘッダーを読み出すためにブート開始時に bootROM によって無条件に発行されるため、必須です。
0x6B	Fast read、24 ビット アドレス、クワッド出力	これは、ブート ヘッダーがクワッド モードで読み出すことができるかをテストするために、ブート ヘッダーをシングル モードで最初に読み出した後、bootROM が発行するコマンドです。読み出しが成功した場合、ブートはクワッド モードで読み出しを継続します。読み出しでエラーが生じると、デュアル モードで読み出しを試みます。
0x3B	Fast read、24 ビット アドレス、デュアル出力	これは、ブート ヘッダーがデュアル モードで読み出すことができるかをテストするために、ブート ヘッダーをクワッド モードで読み出すことに失敗した場合に bootROM が発行するコマンドです。読み出しが成功した場合、ブートはデュアル モードで読み出しを継続します。読み出しでエラーが生じると、シングル モードで読み出しを試みます。

表 4: Zynq UltraScale+ MPSOC のブート コマンド セット、32 ビット アドレス

コマンド コード	コマンド名	詳細
0x13	Read、32 ビット アドレス、シングル出力	このコマンドは、ブート ヘッダーを読み出すためにブート開始時に bootROM が無条件に発行されるため、必須です。
0x3C	Fast read、32 ビット アドレス、クワッド出力	これは、ブート ヘッダーがクワッド モードで読み出すことができるかをテストするために、ブート ヘッダーをシングル モードで最初に読み出した後、bootROM が発行するコマンドです。読み出しが成功した場合、ブートはクワッド モードで読み出しを継続します。読み出しでエラーが生じると、デュアル モードで読み出しを試みます。
0x6C	Fast read、32 ビット アドレス、デュアル出力	これは、ブート ヘッダーがデュアル モードで読み出すことができるかをテストするために、ブート ヘッダーをクワッド モードで読み出すことに失敗した場合に bootROM が発行するコマンドです。読み出しが成功した場合、ブートはデュアル モードで読み出しを継続します。読み出しでエラーが生じると、シングル モードで読み出しを試みます。

Zynq UltraScale+ MPSoC は、ブート中にアドレス サイズをネゴシエートしませんが、バス幅についてはネゴシエートします。表 3 および表 4 に基づいて、128Mb を超えないデータをプログラムする場合に、候補となるシリアル NOR フラッシュがサポートする必要がある最小コマンドは 0x03 です。一方、128Mb を超えるデータをプログラムする場合の最小コマンドは 0x13 です。2 つのシリアル NOR フラッシュをデュアル パラレル トポロジで使用する場合、bootROM ではこれを性能最適化として理解します。したがって、bootROM では、2 つのシリアル NOR フラッシュは両方ともクワッド モードで動作できると見なされるため、より小さいバス幅や混合バス幅はネゴシエートされません。

Zynq-7000 AP SoC は、Zynq UltraScale+ MPSoC と同様に動作しますが、ブート用の 32 ビット アドレスをサポートしていません。つまり、最小コマンド セットの 0x03 を使用する Zynq-7000 AP SoC には表 3 のみ当てはまります。

Zynq UltraScale+ MPSoC と Zynq-7000 AP SoC は両方とも 32 ビット アドレスやクワッド モードを有効にするコマンドを発行しないため、UltraScale、UltraScale+、および 7 シリーズ FPGA と同じです。これらの FPGA では、最大で 2 つの読み出し コマンドを使用 (場合によっては 1 つのみ使用) してプログラミング データ全体のペイロードを読み込む必要があります。

さらに、ユーザーが変更できない bootROM のビヘイビアーと、ロードされるファームウエアやその他のソフトウェアのその後のビヘイビアーの違いを理解することが重要です。ほとんどのデザインでは、ザイリンクスが提供する FSBL (第 1 段階ブートローダー) が最適ソリューションとなり、bootROM によってロードされます。Zynq-7000 SoC 用の FSBL はクワッド出力のシリアル NOR フラッシュで使用でき、クワッド出力コマンドを発行できるため、この場合には最小コマンド セットが 0x03 から 0x03/0x6B に増加します。Zynq UltraScale+ MPSoC 用の FSBL は、さまざまなシリアル NOR フラッシュで使用できる上に、デフォルトでクワッド出力コマンドを発行するので、高い柔軟性を提供します。一方 FSBL はコマンド セットをデュアル出力やシングル出力に削減するコンパイル時間のオプションを提供しますが、このオプションを使用してシングル出力コマンドに減らした場合、最小コマンド セットは 0x03/0x13 として維持されます。

性能

電源投入時のレイテンシに制限があるシステム デザインの場合は、コンフィギュレーションやブート時間をできるだけ短くするために広い帯域幅が必要です。この要件は、バス幅、クロック周波数、そしてこれらを実現するのに必要なザイリンクス デバイスの設定、候補となるシリアル NOR フラッシュの選択肢、さらにはボード設計のあらゆる局面に影響を及ぼします。最適性能を実現するのに必要なものをできるだけ多く (ただし過剰にならないこと) 適用し、最適な性能を実現するように設計することを推奨します。この方法は、低消費電力ソリューション デザインには推奨しません。性能を犠牲にする情報に基づいて判断した方が適切な場合があります。より高い性能を優先する場合は、マージンがなく、部品調達の柔軟性やコスト削減の可能性も制限されます。

バス幅を選択する際には、少ないバス コマンドを使用するソリューションにより部品調達の柔軟性を高めることができます。またクロック周波数を選択する際も、ソリューションの最大値より低い周波数で動作することを選択すると、部品調達の柔軟性を高めることができます。

最大動作周波数を決定するにはタイミング解析が不可欠で、該当するコンポーネントの I/O タイミング (通常はデータシートのパラメーター) やボード デザインの信号遅延 (ボード レイアウトから抽出した情報を使用して IBIS シミュレーション ツールでモデル化) が必要になります。ザイリンクスでは、SPI インターフェイス クロック (SCK) のシグナル インテグリティを確認するためだけでなく、最大周波数を正確に評価できるボードレベルのモデリングを推奨しています。最大動作周波数を把握することで、タイミング バジェット内でスラック値を操作でき、より低速で低性能のシリアル NOR フラッシュとの相互運用性を確保します。

コンフィギュレーションまたはブート開始時、ザイリンクス デバイスはプログラミング データのヘッダーを確実に読み取るために非常に低いクロック周波数を使用します。オプションのクロック周波数の増加はヘッダーを介して伝達され、プログラミング データが生成されるときに設定されます。

UltraScale、UltraScale+、および 7 シリーズ FPGA は、さまざまな周波数オプションを提供しています。最も一般的なオプションは、内部オシレーターから供給される CCLK で、さまざまな分周比をサポートしています。もう一つのオプションは、外部クロック信号から供給される EMCCLK で、2 の累乗での分周をサポートしています。CCLK は EMCCLK より広範な分周オプションを提供し、より高い柔軟性をもたらしますが、EMCCLK ソースの精度によっては、CCLK の方が精度が低くなる可能性があります。一般的に、EMCCLK の使用はコンフィギュレーション時間の精度を向上させるため、または最大周波数に近づけるために特定の周波数に合わせるための性能最適化を目的とします。調達の柔軟性を重視する場合は、システム デザイン要件を満たすことを前提として、高い柔軟性をもたらす CCLK の方が EMCCL よりも適切です。

また、Zynq UltraScale+ MPSoC および Zynq-7000 AP SoC は、さまざまな周波数オプションを提供しています。これらのデバイスのプロセッシング システムは、シリアル NOR フラッシュからの起動に使用される SPI コントローラーを含む豊富なペリフェラル セットでサポートされています。利用できる周波数は高精度で、PLL ソースからの複数の分周ステージによって生成されます。周波数の変更は、ブート ヘッダーに含まれるレジスタ初期化パラメーターを使用して分周器のプログラミングを変更するだけです。

調達の柔軟性を重視した性能選択が可能なタイミングは、最初のデザイン開発期間と量産へ移行する前の段階です。ザイリンクス デバイスはプログラマビリティに優れているため、デザインの変更に遅すぎることはありませんが、提案された変更を実行するには、デザインと Vivado Design Suite の知識を持つ適切な訓練を受けた人材が必要です。デザインを変更すると、会社の方針や認証規則によって再検証が必要になる場合があります。それまでにエンジニアリング チームが解散している可能性があり、極端な場合は、再検証に必要な時間が収益の遅れにつながる可能性もあります。

電源投入シーケンス

電源投入シーケンスでは、ザイリンクス デバイスと候補となるシリアル NOR フラッシュの電源は個別に立ち上がります。各デバイスは電源が有効になると、初期化のためにさらなる時間を要します。ザイリンクス デバイスが先に初期化を完了し、候補となるシリアル NOR フラッシュが読み出しコマンドを受信する準備が整う前にコンフィギュレーション/ブートを開始すると、コンフィギュレーション/ブートはエラーになります。

このためシステム設計者は、電源シーケンス、電源の立ち上がり時間、ザイリンクス デバイスのパワーオン タイミング、およびシリアル NOR フラッシュのパワーオン タイミングの関係を考慮する必要があります。新規デザインでの使用に最適とされる最新のシリアル NOR フラッシュは、電源シーケンス完了後に 1 ミリ秒以内で初期化を完了します。ただし、ボードレベルの電源特性によるタイミング関係によって、ザイリンクス デバイスに対するシリアル NOR フラッシュの準備完了タイミングは大きく影響されます。

シリアル NOR フラッシュのパワーオン タイミングに関する特定要件はありませんが、電源投入シーケンスでは、ザイリンクス デバイスより前にシリアル NOR フラッシュの準備が整う必要があります。シリアル NOR フラッシュに有効な電力が供給された後、初期化時間として少なくとも 1 ミリ秒を割り当てることで、調達の柔軟性を高めることができます。

推奨事項のまとめ

- 調達の柔軟性を高めるためには、必要なコンフィギュレーション性能またはブート性能のみを適用する。可能な限り低い周波数と狭いデータ幅を使用する。
- 開発、デバッグ、プロトタイピングでは、『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 1] に記載されているシリアル NOR フラッシュを選択する。
  - これらの互換性は既に確認されている。
  - ボード固有の電源投入シーケンスは、常に確認が必要。
- 『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 1] に記載されているシリアル NOR フラッシュが量産基準を満たしている場合は、検証プロセスへ進む。
- それ以外の場合は、候補デバイスの互換性を確認してから、信頼できる優れたシステムで検証を実行する。

システム - アプリケーション ストレージとフィールド アップデート

コンフィギュレーションまたはブート後、多くのデザインではコードやデータなどのアプリケーション ストレージとして、動作中にシリアル NOR フラッシュにアクセスします。アプリケーション ストレージは、ファイル システムやフィールド アップデートを可能にするために書き込みと読み出しをサポートする必要があります。信頼性の高いフィールド アップデート機能をサポートするために、2 つのプログラミング データ ペイロードが格納されます。1 つは、ほとんど書き換えられることのないゴールデン イメージで、もう 1 つはフィールド アップデートのときにすべて書き換えられることが多いアップデート イメージです。

UltraScale、UltraScale+、および 7 シリーズ FPGA では、コンフィギュレーション完了後に SPI インターフェイスへ直接アクセスをサポートしています。これにより、SPI コントローラー機能を実装するソフト IP コアの接続や独自アプリケーション用のカスタム ロジックの接続が可能になります。Zynq UltraScale+ MPSoC および Zynq-7000 SoC には、SPI コントローラー機能が統合されているため、ブート後にも引き続き SPI インターフェイスを使用できます。

システム デザインでは、動作時におけるフラッシュの読み出しおよび書き込みアクセスを管理します。この広範なステートメントは、SPI コントローラー機能をサポートするだけでなく、ファームウェア、デバイス ドライバー、アプリケーション、さらにはオペレーティング システムもサポートします。オペレーティング システムには、ザイリンクス デバイス上で実行するものとリモートで実行するもののほか、ザイリンクス デバイスの SPI インターフェイスで操作を制御するものがあります。コマンド、データ幅、およびクロック周波数は、システム デザインの判断でボードレベル ソリューションの最大能力まで使用可能です。調達の柔軟性を重視する場合、基本的なガイダンスはコンフィギュレーションやブート用のシリアル NOR フラッシュの動作時の使用と同じで、必要なものをできるだけ多く (ただし過剰にならないこと) 適用することです。

システム デザインの複雑性、サードパーティの大規模ブロックやレガシ デザイン ソース (ハードウェア/ソフトウェア) の日常的な再利用を考えると、特定のコマンド、データ幅、およびクロック周波数の使用を指示することは不可能です。メリットをもたらす比較的実用的なアプローチは、使用するコマンド、データ幅、およびクロック周波数をすべて把握した上で開発を進めることです。

- SPI インターフェイスとシリアル NOR フラッシュは、いずれも制限値を超えて動作しないようにする。
- 最終的には排除する目的で、あまり一般的ではないコマンドまたはベンダー固有のコマンドの使用を特定する。
- ベンダー固有のステータス、コマンド、識別レジスタの不要な使用を特定する。
- できるだけ周波数やバス幅を削減し、不要な性能を特定する。

このアプローチは、ハードウェア設計者とソフトウェア設計者の両方が関わることになるため、特に複雑なシステム デザインをチーム ベースで進める際には多くの専門分野にまたがります。ある 1 つのプロジェクトに関わるすべての開発者が代替設計案を持つように求めるのは現実的ではありませんが、開発ガイドラインに記載されている事項に遵守することは難しいことではありません。

システム デザインのその他の部分では、動作時、コンフィギュレーション、またはブート時にシリアル NOR フラッシュのアプリケーションを分割できます。

- デザイン セキュリティに対する要求は一層高まるため、シリアル NOR フラッシュのベンダー固有機能を使用するアドホック ソリューションよりも、ザイリンクス デバイスの暗号化、認証、およびセキュア ブートで対応する必要があります。ザイリンクス デバイスはさまざまなセキュリティ機能をサポートしています。これらのセキュリティ機能の多くはデバイスに統合されており、キーや設定に不揮発性の eFUSE を使用することで、開発コストや BOM コストを増加させることなく信頼できる保護機能を提供できます。
- オープンソース ソフトウェアの使用がますます増加するため、選択時の注意事項として、アップストリーム開発の健全性を評価することが重要です。ドライバー、ファイル システムおよび第 2 段階ブートローダーのソースは、将来的に利用できる NOR フラッシュでの置き換えを可能にする (または制限する) ため、特に重要です。開発コミュニティの動向は、JFS2 から UBIFS へ移行するファイル システムで確認できます。

使用している容量のシリアル NOR フラッシュの形状、適合性、および機能を置き換える場合、さらに大容量のシリアル NOR フラッシュを選択することが可能ですが、この場合はコストが増加します。もともとの容量のフラッシュが一時的に使えなくなっていたり、製造が中止された場合は大容量への移行を考えることができますが、その他の場合はあまり好ましくありません。製造中止のリスクを抱えるよりも、一時的あるいは恒久的にコストが増加する方がよい場合もあります。この理由から、必要以上に高密度なシリアル NOR フラッシュ デバイスを使用していないかを確認し、そのようなシステム ビヘイビアーの実装を回避してください。

推奨事項のまとめ

- 調達の柔軟性を高めるためには、必要なランタイム機能および性能のみを使用する。
  - 使用されているコマンドを認識し、削減および記録する。
  - 可能な限り低い周波数と狭いデータ幅を使用する。
- システムの寿命期間中に広範なシリアル NOR フラッシュに対応できるようにデザインする。
  - ベンダー識別レジスタの不要なソフトウェア チェックを排除する。
  - できるだけ多くの機能をハードウェアからソフトウェアへ移行する。
  - 長寿命のためには、健全なアップストリーム開発が行われているソフトウェア ソースを選択する。

プログラミング - メモリ コンフィギュレーション ファイル

Vivado Design Suite は、さまざまな設計フローに対応しています。シリアル NOR フラッシュがコンフィギュレーションやブートに使用される場合、これらのフローはプログラミング データの生成完了後、同じフローになります。コンバージェンスのポイントは、シリアル NOR フラッシュのプログラミングに使用されるメモリ コンフィギュレーション ファイルです。メモリ コンフィギュレーション ファイルは、シリアル NOR フラッシュにプログラムされるデータを含む単なるファイルです。1 つまたは複数のプログラマブル ロジック ビットストリーム、プロセッシング システム コード、またはデータ パーティションが含まれ、その他にアプリケーション ストレージ、ファイルシステム、オペレーティング システム イメージ用にさらなるパーティションを含めることも可能です。

メモリ コンフィギュレーション ファイルには、MCS、HEX、SREC などの業界標準のファイル形式がいくつかあります。これらの形式は ASCII テキスト レコードに基づいており、これはディスク上のファイル サイズが 2 倍以上に増加する表現形式ですが、不連続データ、ブロック、チェックサムをサポートし、読みやすく編集も簡単です。BIN や DAT のファイル拡張子を持つバイナリ データ ファイルも、余計なものが付いていない表現方法として適しています。Vivado Design Suite では、MCS と BIN 形式をサポートしています。

プログラミングに Vivado Design Suite を使用

量産向けに同じシリアル NOR フラッシュを使用する場合には、検証プロセスで引き続き Vivado Design Suite を使用できるため、開発、デバッグ、プロトタイピングの段階で『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 1] に記載されているシリアル NOR フラッシュを間接プログラミングする際は Vivado Design Suite を使用することを推奨しています。Vivado Design Suite のウィザードが、メモリ コンフィギュレーション ファイルの生成プロセスから、その後のシリアル NOR ファイルのプログラミングまでをガイドしてくれます。

UltraScale、UltraScale+、および 7 シリーズ FPGA の場合は、Vivado GUI の [Open Hardware Manager] → [Generate Memory Configuration File] をクリックして、『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 1] に記載されているシリアル NOR フラッシュを選択できます。ウィザードに従って、データ バス幅を選択し、プログラマブル ロジック ビットストリームとメモリ コンフィギュレーション ファイルに含めるその他のデータを指定します。[Generate Memory Configuration File] の出力は MCS ファイル形式であり、`write_cfgmem` ユーティリティで作成されます。Vivado Design Suite のハードウェア マネージャーを使用してシリアル NOR フラッシュに MCS ファイルのコンテンツを間接プログラミングする方法は、『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 1] を参照してください。

Zynq UltraScale+ MPSoC および Zynq-7000 AP SoC の場合は、ソフトウェア開発キット (SDK) GUI の [Create Boot Image] ダイアログボックスでブート イメージ パーティションやメモリ コンフィギュレーション ファイルに含めるその他のデータを指定できます。[Create Boot Image] の出力は MCS ファイル形式であり、bootgen ユーティリティで作成されます。SDK を使用して、シリアル NOR フラッシュに MCS ファイルのコンテンツを間接プログラミングする詳細は、『Using Xilinx SDK』(UG782) [参照 8] を参照してください。『Xilinx Software Command-Line Tool (XSCT) Reference Guide』(UG1208) [参照 9] には、[Xilinx Software Command Line Tool] を起動して同様の動作をスクリプト化するユース ケースがあります。Bootgen の詳細は、『Zynq UltraScale+ MPSoC ソフトウェア開発者向けガイド』(UG1137) [参照 10] および 『Zynq-7000 SoC ソフトウェア開発者向けガイド』(UG821) [参照 11] も参照してください。その他にもザイリンクスではエンベデッド デザイン チュートリアルとして、『Zynq UltraScale+ MPSoC: エンベデッド デザイン チュートリアル』(UG1209) [参照 12] および 『Zynq-7000 SoC: エンベデッド デザイン チュートリアル』(UG1165) [参照 13] を提供しています。

明白には示していませんが、ハードウェア マネージャーと SDK は両方ともシリアル NOR フラッシュのデータ アレイだけでなく、その他にもプログラムする必要があります。データ アレイに MCS ファイルのコンテンツをプログラムすることに加え、必要に応じてプログラミング ツールで不揮発性制御レジスタの特定ビットをプログラムします。これらの不揮発性制御レジスタのビットは、ベンダー固有またはデバイス固有である可能性があり、32 ビット アドレスやクワッド モードなど、シリアル NOR フラッシュの機能を恒久的に有効にするためのプログラミングが必要です。ザイリンクス デバイスは、コンフィギュレーションやブート中に機能を有効にするためのコマンドを発行しないため注意が必要です。シリアル NOR フラッシュは、これらの機能が有効に設定されている状態で電源が投入される必要があります。

プログラミングにサードパーティ ソリューションを使用

『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 1] に記載されているシリアル NOR フラッシュの間接プログラミングに Vivado Design Suite を使用して開発、デバッグ、プロトタイピングを行った場合は、量産用に同じシリアル NOR フラッシュを使用する場合の検証プロセスにも Vivado Design Suite を使用できます。量産基準に対応するために別のシリアル NOR フラッシュを使用する場合は、検証プロセスにサードパーティの直接インシステム プログラミング ソリューションを使用する必要があります。いずれの場合も量産開始後、ボード上にプログラム済みのシリアル NOR フラッシュを直接組み立てるためには、低コストで大量にできる方法としてサードパーティの直接バルク プリプログラミング ソリューションの使用が推奨されています。

MCS は業界標準のファイル形式であるため、ほとんどのサードパーティ プログラミング ソリューションは Vivado Design Suite で生成された MCS ファイルを直接読み込みます。Vivado Design Suite は BIN ファイルも生成できます。通常、サードパーティ プログラミング ソリューションにはファイル変換ツールがあります。また SourceForge の SRecord プロジェクトなど、優れたオープンソース ファイル変換ツール [参照 14] も利用できます。

サードパーティのプログラミング ソリューションを使用する場合、ザイリンクス デバイスのコンフィギュレーションやブートをサポートする目的で 32 ビット アドレスやクワッド モードなどの機能を必要に応じて恒久的に有効にするために、シリアル NOR フラッシュの不揮発性制御レジスタのビットをプログラムする必要があります。不揮発性制御レジスタ ビットのプログラム自体は簡単ですが、次の注意点があります。

- プログラミング プロセスでは明白な手順ですが、見落とす可能性があります。
  - 直接インシステム プログラミングを使用する場合の検証時の補正作業は、プログラミング プロセス中に忘れた手順を適用するだけです。
  - 直接バルク プリプログラミング時の補正も同様ですが、2 回目のプログラミング パスには直接コストがかかります。
- サードパーティのプログラミング ソリューションで各消去が実行された場合、その後にプログラミングが必要になることがあります。これはベンダーによってはデータ アレイのみが消去されるところ、一部のベンダーの消去デバイスはデータ アレイと不揮発性制御レジスタの両方を消去するためです。

したがって、Vivado Design Suite からサードパーティ プログラミング ソリューションへ移行する際は、新しいプログラマの最初の項目を常に確認する必要があります。特にバルク プリプログラミングでは、さらなるプログラミング パスを回避するために重要です。

推奨事項のまとめ

- 設計期間の短縮、リスク軽減、広範なエンジニアリング アクセスを可能にするためには、プロジェクトの早期段階で『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 1] に記載されているシリアル NOR フラッシュの間接プログラミングに Vivado Design Suite を使用する。
- 『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) に記載されているシリアル NOR フラッシュを引き続き使用する場合、または量産基準に対応するためにさらなる互換オプションとしてフラッシュを置き換える場合、のいずれかの好みに応じて検証プロセスや量産開始時にサードパーティ プログラミング ソリューションを使用する直接プログラミングへ移行する。
  - サードパーティのプログラミング ソリューションで、既存の MCS または BIN ファイルを引き続き使用する。
  - ザイリンクス デバイスがコンフィギュレーション/ブート中に 32 ビット アドレスやクワッド モードなどのシリアル NOR フラッシュの機能を使用する場合は、サードパーティ プログラミング ソリューションを使用して、これらの機能を恒久的に有効にしておく。

製造 - コンポーネントの交換

ほとんどの企業で、ビジネスはスタッフが機能的に組織されたチームによる取り組みによって成り立っています。あるビジネスの全側面を一人の個人がすべて把握して判断することはありません。時間が経過するにつれて組織やスタッフは変わります。継続的な製品計画がなければ、企業は徐々にシステムを維持およびサポートする能力を失うことになります。これは、シリアル NOR フラッシュの置き換えに対応するアジリティの喪失につながります。

通常、システムの部品管理 (BOM) において調達状況の変化に対応する責任は、部品調達/製造チームにあります。当然のことながら、これらのチームは既存の設計チームよりもシステムの設計に関する知識はありません。それにもかかわらず、部品調達チームは次の作業を行う責任があります。

- シリアル NOR フラッシュの形状、適合性、および機能の代替品を特定する
- 最低 1 つ合格するまで、ラボ内で代替品を検証する
- ECO (Engineering Change Order) を作成する

これらの作業は、既存の設計チームから情報を取得しなければ、膨大な時間とリスクが生じます。一方、既存の設計チームは解散している可能性があります。つまり、失われた情報を再度取得するために自社システムをリバース エンジニアリングしなければならない状況が簡単に想像できます。

設計チームには、開発中に製造チーム向けの資料を作成するように促します。これらの情報は、文書管理システムで管理し、情報が必要となるすべてのチームがアクセスできるようにします。シリアル NOR フラッシュについては、このアプリケーション ノートに記載されている考慮事項の概要が、記録する内容を知る上で役立ちます。さらに、代替製品のプログラミングと検証の手順を作成して文書に記録します。プロセスに精通していないスタッフにクリーンルーム リハーサルとして支援なしに手順を実行させて、資料の完全性をテストします。

推奨事項のまとめ

- 開発チームにシステムに関する文書を作成させる。
  - シリアル NOR フラッシュの機能要件および性能要件を取得する。
  - 代替製品のプログラミングと検証の手順を作成する。
- 文書管理機能を使用して作成した文書を管理する。

まとめ

これまでの 10 年間で NOR フラッシュ市場は大きく変化し、コスト削減への要求が高まる中、多くのプロジェクトでは代替設計案の採用が不可欠になりました。この考え方は、中/長寿命の製品を不安定なコンポーネント市場から守ることができ、また厳しいコスト目標を持つ製品には、計画的かつ便宜的なコスト削減の可能性をもたらします。

このアプリケーション ノートで提供する推奨事項は、NOR フラッシュを使用する場合の一般的な課題への対処方法、およびコスト削減のための有効な方法を示しています。さらに重要な点は、製品に採用する NOR フラッシュの選択は調達の柔軟性に大きく影響を与える可能性があるため、情報に基づいた意思決定と事前計画がリスク削減と収益性向上に役立つことを説明しています。

Xilinx Documentation Navigator およびデザイン ハブ

Xilinx Documentation Navigator (DocNav) では、ザイリンクスの資料、ビデオ、サポート リソースへアクセスでき、特定の情報を取得するためにフィルター機能や検索機能を利用できます。Xilinx Documentation Navigator を開くには、次のいずれかを実行します。

- Vivado IDE で [Help] → [Documentation and Tutorials] をクリックします。
- Windows で [スタート] → [すべてのプログラム] → [Xilinx Design Tools] → [DocNav] をクリックします。
- Linux のコマンド プロンプトに「docnav」と入力します。

ザイリンクスのデザイン ハブでは、資料へのリンクがデザイン タスクおよびトピックごとにまとめられており、これらを参照することで重要なコンセプトに関する知識を得たり、よくある質問 (FAQ) を参考に問題を解決できます。デザイン ハブにアクセスするには、次のいずれかを実行します。

- Xilinx Documentation Navigator で [Design Hubs View] タブをクリックします。
- ザイリンクス ウェブサイトのデザイン ハブ ページを参照します。

注記: Xilinx Documentation Navigator の詳細は、ザイリンクス ウェブサイトの Documentation Navigator ページを参照してください。

参考資料

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. 『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908: 英語版、日本語版)
2. エンドユーザー使用許諾契約
3. DediProg (www.dediprog.com)
4. TotalPhase (www.totalphase.com)
5. Corelis (www.corelis.com)
6. Pomona Electronics (www.pomonaelectronics.com)
7. 3M (www.3m.com)
8. Using Xilinx SDK (UG782)
9. Xilinx Software Command-Line Tool (XSCT) Reference Guide (UG1208)
10. 『Zynq UltraScale+ MPSoC ソフトウェア開発者向けガイド』(UG1137: 英語版、日本語版)
11. 『Zynq-7000 SoC ソフトウェア開発者向けガイド』(UG821: 英語版、日本語版)
12. 『Zynq UltraScale+ MPSoC: エンベデッド デザイン チュートリアル』(UG1209: 英語版、日本語版)
13. 『Zynq-7000 SoC: エンベデッド デザイン チュートリアル』(UG1165: 英語版、日本語版)
14. SRecord Package (www.srecord.sourceforge.net)

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
2018 年 3 月 5 日	1.0	初版

お読みください: 重要な法的通知

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@xilinx.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。